PROJECTS THEORY APPLICATIONS CIRCUITS TECHNOLOGY

NUTS AND VOLTS

EVERYTHING FOR ELECTRONICS

www.nutsvolts.com
March 2014

Solar Charge CONTROLLER

Easy to build ✶
Inexpensive ✶
Charges up to 12 amps ✶

✦ **Breaking The Arduino Speed Limit**

How fast can you push the Arduino platform?

~~Kind of Fast~~

~~Pretty Fast~~

✓ **VERY FAST**

NetBurner Developer Training

Las Vegas Nevada

April 3 – 4, 2014

Learn how to add network capability to your new or existing products!

Two-day seminar includes:

› Development kit
› Training materials
› Lunch (both days)

This developer event features highly technical, in-depth sessions focused on building web and network applications. You will learn about our next-generation hardware and software such as NetBurner's Eclipse IDE, OS, TCP/IP stack with the latest NetBurner network core modules.

How to Register

Phone: Call 1-800-695-6828 to register for the NetBurner Developer Training Seminar.
Online: To register online today, please visit http://www.netburner.com/training2014

NetBurner
Networking in One Day!

Information and Sales | sales@netburner.com
Web | www.netburner.com
Telephone | 1-800-695-6828

freescale™
Alliance Member

Fun starts behind a 5" screen

price: $249⁰⁰

mikromedia 5 for Tiva™

ARM Cortex™-M4F on-board

Powerful Workhorse

Mighty TM4C129xNCZAD rocks: 120 MHz operation, 1 MB of Flash, 256 KB of SRAM, μDMA controller, on-chip LCD controller and a lot more.

Size does matter

5" Touch Screen

Huge touchscreen in 800x480px resolution brings awesome graphics and vivid colors. You've never seen such a big display driven just by a microcontroller.

Communicate wirelessly

CC3000 WiFi module

Internet of things - check! Insanely popular CC3000 WiFi module is right here for you. It features easy to use SimpleLink IPv4 TCP/IP stack.

Beyond expectations

Debugger on board

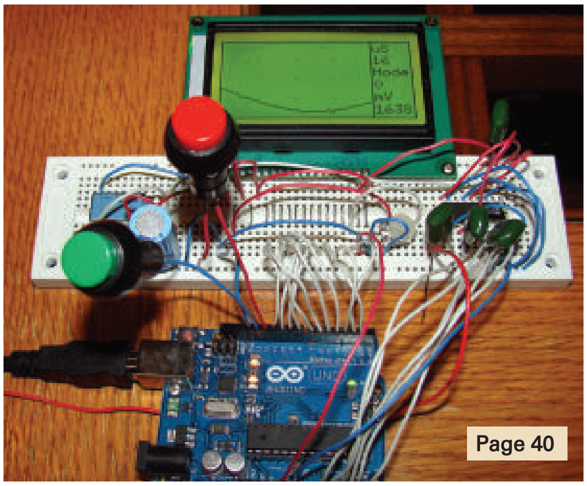
For the first time, we're embedding programmer and debugger on mikromedia board. Just plug in the USB cable and there you go!

MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

GET IT NOW
http://www.mikroe.com/store/

March 2014

Page 33



Nuts & Volts (ISSN 1528-9885/CDN Pub Agree #40702530) is published monthly for $26.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL MAILING OFFICES. POSTMASTER: Send address changes to **Nuts & Volts, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@nutsvolts.com.

Next Gen Data Logging Welcome to DAQ 2.0

Use Java™ and Android™
for the Coolest DAQ Apps

Wireless DAQ/Tablet Bundle

- BTH-1208LS multifunction Bluetooth™ DAQ device
- Google™ Nexus™ 7 Tablet included
- UL for Android Library
- Example programs
- Ready-to-run Demo Apps available at Google Play™

Collect data wirelessly with the BTH-1208LS DAQ device

Best Deal of the Year – *Only* $429

mccdaq.com/BTHBundle

MEASUREMENT COMPUTING™

The Value Leader in Data Acquisition

Contact us
1.800.234.4232

DEVELOPING PERSPECTIVES

Know Your Basics

One of my ongoing activities is working with a DoD-funded research team that is developing an open source model of the human body. In a few years, a professor should be able to teach medical students how, for example, the lungs work without sacrificing an animal. The interesting point about this project — which involves computer scientists, engineers, physicians, and physiologists — is that the common language is simple discrete component electronics.

For example, in developing a model of the lungs, we represent each lung with a diode, a few resistors, and a capacitor. That's it. Add AC or DC driving signals, and the current and voltage swings mimic the pressures and flows in the lungs.

The take-away of this illustration is that it's important for you to learn the basics. I'm talking Ohms Law, serial and parallel discrete components, and simple signal sources. This might seem self-evident, but since the introduction of the increasingly popular microcontrollers and standard sensors and effectors, it's possible to create electronic devices without ever touching a capacitor or resistor. Why use a pull-up resistor when microcontrollers (such as the Arduino) allow you to specify pull-ups in software?

Of course, if your time is limited and you have a specific project in mind, you want a solution as soon as possible. However, if your goal is to master the art of electronics, then you need to understand the basics. Ten years from now when the current generation of microcontrollers — and your knowledge of their specifics — is worthless, there will be applications for Ohms Law and basic circuitry. As illustrated by my experience working with scientists from varied backgrounds, basic circuitry can be a Rosetta Stone for communications — second only to pure mathematics.

So, let's say you're sold on the concept of getting a solid foundation in the basics. Just how do you get this grounding? Well, in addition to the occasional introductory articles in *Nuts & Volts*, check out the classics such as one of the introductory texts from Forrest Mims III (**www.forrestmims.org**). Then, there's the timeless *Art of Electronics* by Horowitz and Hill. If you're not into reading, there are dozens of introductory electronics tutorials on YouTube.

These passive sources of information are all perhaps necessary, but by no means sufficient to get you where you need to be. You need some hands-on experience to ground your theoretical understanding of basic electronics. Pick up a kit that uses discrete components — one that lets you easily substitute the components. Another route is to tear down every electronic device you can get your hands on.

Don't let any electronic device that is destined for the landfill escape your pliers and soldering iron. Take notes and take hostages (remove components for repurposing). Try to figure out the underlying circuit and create a schematic. Then, try to improve on the basic circuit design.

Once you've achieved this level of success, you've mastered the basics. **NV**

ROS-Training

SPECIAL OFFER: 20% OFF YOUR FIRST CLASS!
To reserve your discount sign up at
www.ros-training.com/nuts_volts

LEARN • BUILD • SHARE
www.ros-training.com

NUTS AND VOLTS
EVERYTHING FOR ELECTRONICS

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct.
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore orders only **1-800-783-4624**
www.nutsvolts.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

FOUNDER
Jack Lemieux

PUBLISHER
Larry Lemieux
publisher@nutsvolts.com

ASSOCIATE PUBLISHER/ ADVERTISING SALES
Robin Lemieux
robin@nutsvolts.com

EDITOR
Bryan Bergeron
techedit-nutsvolts@yahoo.com

CONTRIBUTING EDITORS
Joe Pardue, John Gavlik, Craig Lindley, Martin Hebel, Bob Davis, John Mastromono, Ron Newton, Fred Eady, Jon McPhalen, Jeff Eckert, Paul Verhage

CIRCULATION DEPARTMENT
subscribe@nutsvolts.com

SHOW COORDINATOR
Audrey Lemieux

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@nutsvolts.com

WEB CONTENT
Michael Kaudze
website@nutsvolts.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION
Sean Lemieux

READER FEEDBACK

WARNING SYSTEM Feedback

Regarding Roger Secura's "Build a Wireless Silent Alarm Warning System" article in the December 2013 issue: This article was superb. The graphics were great, it was very complete with completely descriptive text, BOM, sources, a plain English picture of the installation, a plain English view of the components, and very clear circuit diagrams. This article should be held up as a standard to which all articles should be written.

Bruce Pease

Regarding Roger Secura's December issue project and the TWS434A and RWS434 parts.

I went to **www.rentron.com** to obtain parts as directed by the article. The website is no longer in service and the phone number is disconnected. I tried several parts companies for the two parts, but no luck. Anyone locate a source?

Bill White

I found some suppliers that carry the TWS434A and RWS434 parts; see the list below.

Roger Secura

OPTION #1
https://ishtroni.ipower.com
Voice USA: 770-590-1822
Sales: sales@ishtronics.com
TWS434A Transmitter = $7
RWS434 Receiver = $7

The website is a little confusing, so follow these instructions:

1. Click on the 'Products' link at the top of the page.

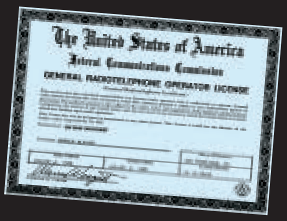
2. Scroll to the bottom of the page and click on the link 'RF ASK

Continued on page 72

EARN MORE MONEY
Get your dream job!
Be an FCC Licensed Wireless Technician!
Make up to $100,000 a year and more with NO college degree

Get your "FCC Commercial License" with our proven Home-Study Course!

- No costly school. No classes to attend.
- Learn at home. Low cost!
- No experience needed!
- MONEY BACK GUARANTEE: You get your FCC License or money refunded!


The United States of America
Federal Communications Commission
GENERAL RADIOTELEPHONE OPERATOR LICENSE

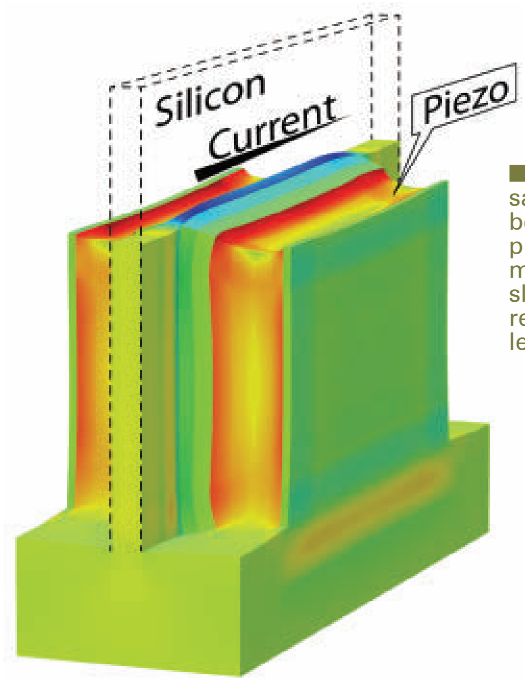
Move to the front of the employment line in Radio-TV, Communications, Avionics, Radar, Maritime and more... even start your own business!

Call now for FREE info: **800-877-8433** ext. 109
www.LicenseTraining.com
COMMAND PRODUCTIONS FCC License Training
Industrial Center, 480 Gate Five Rd., PO Box 3000, Sausalito, CA 94966-3000

■ BY JEFF ECKERT

Post comments on this article at **www.nutsvolts.com/index.php?/magazine/article/march2014_TechKnow14**.

ADVANCED TECHNOLOGY



■ Silicon is sandwiched between piezoelectric material slices to reduce leakage.

Squeezing Out the Leakage

As is commonly known, all silicon transistors drain a device's batteries by using both active power and leakage power. It is not so widely appreciated that leakage has become a major concern for IC designers who work with submicron (65 nm and below) process technologies. In fact, leakage can account for as much as 30 to 50 percent of total IC power consumption which is one reason why you have to charge your laptop and cell phone so often.

If you compress a vein in your arm, this restricts the flow of blood. Oddly enough, the opposite is true of silicon: The more you compress it, the more easily electrons can move through it. This is why transistors in modern microchips are continuously exposed to pressures of up to 10,000 ATM (nearly 147,000 PSI). Unfortunately, squeezing the semiconductor also increases its leakage current.

However, Tom van Hemert and Ray Hueting of the Netherlands' University of Twente (**www.utwente.nl**) have recently theorized that you can beat the system by sandwiching the silicon material between two layers of a piezoelectric material. Because this material expands only when the device is in the on state, the pressure and leakage will slack off in the off state.

At least in theory, a transistor of this type can operate on a charge of 50 mV rather than the standard 60 mV, offering you the choice of reduced leakage or a higher on-state current. Details were published in a recent issue of *IEEE Transactions* on electron devices. ▲

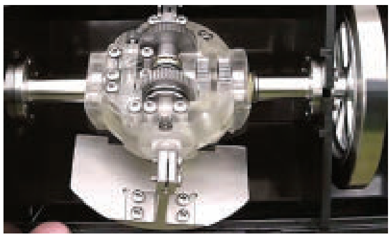
From Chaos to Electricity

Around 2008 or 2009, a civil engineer by the name of Martin Wickett was thinking about what he could possibly use to drive a generator and came up with an answer: "whatever." It was then that the Whatever Input to Torsion Transfer (WITT) transmission was born. In a nutshell, the WITT is a mechanism that converts energy collected by two pendulums into unidirectional rotation in a flywheel, which then drives a generator. Thus, random movement of the WITT is converted to electricity.

Having undergone a few years of development and receiving financial and engineering support from a consortium of universities and other participants, a working device has now been completed by Supacat Ltd. (**www.supacat.com**) — a Devon, UK based developer of high mobility vehicles. It is slated to undergo extensive testing by the University of Exeter at their Dynamic Marine Component Test facility. Because things that float around in the sea are exposed to substantial wave movement, it should be the perfect environment for evaluating the device's performance.

The current version weighs about 100 kg (220 lb), is built from precision-engineered components and cast aluminum, and is about the size of a desktop computer. The beauty of it is that the WITT can be scaled up or down for a nearly unlimited range of applications. According to Paul Weston, renewable energy technical manager for ship repair and conversion company A&P Falmouth, "The device can also be used in all types of movement whether on land or at sea, on a backpack, yacht, or ship. It is a pioneering project that transfers motion into energy, and we are delighted to be involved with it."

If you want to see one in operation, check out the videos at **www.witt-energy.com**. ▲



■ The WITT generates electricity from random motion.

COMPUTERS and NETWORKING

Mac Pro — Small Package, Big Price

Since the introduction of the Power Mac G5 in 2003, Apple has been using essentially the same huge (approx. 20 x 8 x 19 in, or 51 x 20.5 x 47.5 cm), rectangular aluminum case. Desktop Macs have also been almost ridiculously heavy. The 2012 Mac Pro 12-core model, for example, tipped in at nearly 40 lb (18 kg). All that changed last year.

The late 2013 12-core machine is cylindrical, with a height of 9.9 in (25 cm) and a diameter of 6.6 in (16.7 cm); its weight has been reduced to just 11 lb (5 kg). Opinions are mixed about its appearance (which is not as dark as it looks in publicity photos), but it does look like something that you would want to keep on the desktop instead of parking under your desk.

The size reduction is accomplished in a way that any given user may or may not appreciate. The old units had lots of space inside for hard drives, optical drives, and a bunch of expansion cards, but the new design is geared more for external expansion via connectors and ports. The rear panel provides four USB 3.0 and six Thunderbolt 2 ports, plus dual gigabit Ethernet and an HDMI 1.4 UltraHD connection. And, of course, there is wireless access via 802.11AC and Bluetooth 4.0. It also saves space by eliminating the use of mechanical hard drives, instead providing up to 1 TB of PCIe-based Flash storage.

■ Mac Pro with covers off, core exposed.

In terms of processing power, you can choose the paltry pair of 3.7 GHz quad-core Xeon E5s or max it out with twin 2.7 GHz six-core processors. It can be configured with up to 64 GB of DDR3 RAM. Apple says that will give you up to 7 Tflops of performance, compared to a measly 2.7 Tflops in the previous models.

Here's the nitty gritty. Mac Pros have never been for the budget-minded, and that hasn't changed. The eight-core machine starts at $2,999 and the 12-core one at $3,999. If you max out the RAM, Flash storage, and graphics, you're staring at $9,599, and that doesn't even include a mouse or keyboard. And, you really need to think about the Sharp 32 in 4K display ($3,595), right? But, hey. You don't really need that new car, do you? Details at **www.apple.com/mac-pro**. ▲

Faster, Cheaper SSD

If you're looking for a storage upgrade, you may be pondering the choice between a new hard drive and a solid-state drive. The SSD has lots of advantages, such as faster bootup, less power draw, no moving parts, and so on. However, they have one big disadvantage: price. You can pick up a 1 TB Seagate HDD for about $120, whereas a 600 GB Intel 320 Series SSD is going for almost ten times as much. Now, Samsung (**www.samsung.com/us**) has at least narrowed the gap with the 840 EVO which features the mSATA form factor, making it about a quarter of the size of a 2.5 in SSD. It is thus suitable not only for desktops but for ultra-slim notebooks, as well.

■ Samsung's 840 EVO SSDs offer faster data transfer, somewhat lower price.

The 840 EVO is built on Samsung's 128 Gb NAND Flash memory, and the company has produced a 1 TB version by packing in four Flash memory packages — each with 16 layers of memory chips. What often separates a top-notch SSD from a mediocre one is the controller technology, and this one uses the company's fifth-gen MEX 5 multicore controller which runs at 400 MHz — a third faster than the previous generation. This gives you read/write speeds of 540/520 MB/s. That compares nicely to the Seagate's maximum sustained transfer rate of 175 MB/s.

You'll still have to shell out some bucks, but as of this writing, at least one online vendor is offering the 1 TB version for $529. You can save money by scaling back to 750 GB ($429), 500 GB ($309), or even 250 GB ($169.99). ▲

CIRCUITS and DEVICES

3D Scanner Unveiled

So, you've been thinking hard about getting a 3D printer but don't really know how to create the files that they use. One bit of good news is that Adobe has added 3D printing to the Photoshop Creative Cloud (but apparently not the Creative Suite). So, what if you don't subscribe to the Cloud, or you aren't particularly adept at Photoshop? What if you just want to make copies of things that are lying around in your house? Well, check out Matterform, billed as "the world's first affordable high-resolution 3D scanner."

According to co-inventer Adam Brandejs, "We thought it would be popular with the hacker/maker crowd, but the applications are much broader than that. We've had interest from designers, artists, archeologists, dentists, and even parents who want to scan their kid's artwork."

The Matterform is made up of a moving HD camera head with dual lasers and a rotating platform. It comes fully assembled, so all you have to do is place an object on the platform, push a button, and let it do the rest. Scanned items can be up to 9.8 in (25 cm) high, 7 in (18 cm) in diameter, and 6.6 lb (3 kg) in weight. The scanner can handle details as small as 0.02 in (0.43 mm), and a full high-res scan takes less than 10 min. You then can export the file in STL, OBJ, or PLY format. Matterform supports Windows 7+ and Mac OS 10.7+ platforms. You can buy one online at **www.matterform.net** for $579. ▲

■ The Matterform 3D scanner captures an object in less than 10 minutes.

It's a Real Crock

If you thought it would take a while before the Internet of Things got out of hand, it may be time to rethink that. First of all, recall that the folks at Belkin introduced the WeMo home automation line (**www.belkin.com/wemo**) a year or so ago which consists of a range of devices that can be controlled from anywhere via an iOS or Android. For example, you can replace a wall switch with the WeMo Light Switch, enabling you to turn a light on or off from anywhere in the world, as long as your home is equipped with a Wi-Fi network and you have a 3G or 4G LTE connection on your smartphone or tablet. Or, you can replace an AC wall outlet with the WeMo Switch and get remote control of TVs, lamps, fans, and so forth.

The company may have gone a bridge too far, however, by partnering with Jardin Corporation — hawker of more than 120 brands of consumer products including Mr. Coffee, First Alert, Oster, and (most pertinently) Crock-Pot. The first product to emerge from the partnership is the WeMo-enabled Crock-Pot Smart Slow Cooker which lists at $99.99 (as opposed to $29.99 for a five-quart manual model). As "the first smartphone-controllable slow cooker, it allows you to remotely adjust the cooker's settings, receive reminders from it, change the cooking time, and so on.

Seriously now. Do you really want to do that? Isn't the point of a slow cooker that you can throw in a bunch of ingredients, completely ignore the appliance all day, and come home to an overcooked, mushy, nasty tasting dinner? ▲

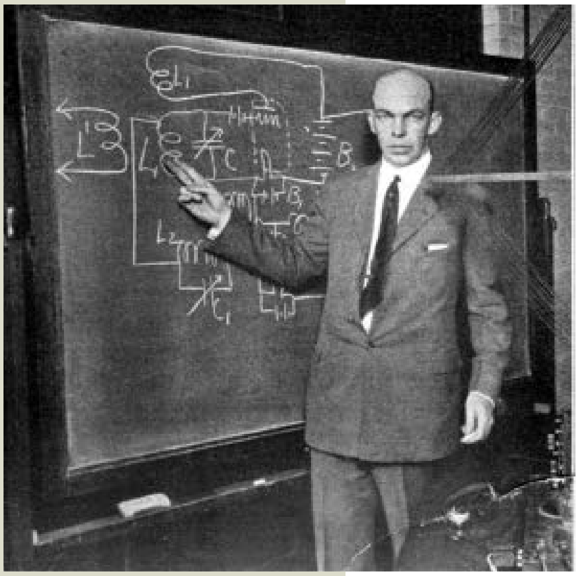
■ The WeMo-enabled Crock-Pot®, controllable via iOS or Android.

INDUSTRY and the PROFESSION

100th and 60th Anniversaries

One hundred years ago while an undergraduate at Columbia University, Edwin Howard Armstrong invented and patented the regenerative circuit which allows a signal to be amplified many times by the same vacuum tube or other component. He followed this up with the superheterodyne receiver in 1918, and the super-regenerative circuit in 1922. Perhaps most notably, he was also the inventor of frequency modulation which, of course, remains the basis of today's FM radio.

In 1934, Armstrong accepted an offer from RCA President David Sarnoff to work for him. According to accounts, Sarnoff was impressed with the FM concept but soon realized that it threatened RCA's AM empire. Plus, the company was more interested in getting into television broadcasting. Taking the bull by the horns, Armstrong personally financed construction of the first FM station, and the early FM industry began to take off. Unfortunately, Sarnoff (with encouragement from AT&T) sabotaged it by convincing the FCC to reallocate the FM spectrum, thereby rendering all existing FM receivers useless. In addition, RCA claimed that it had invented FM and was issued its own patent on the technology, leading to a legal battle that continued for years. This left Armstrong in extreme financial and emotional trauma.


■ Edwin Armstrong explains the super-regenerative receiver, 1922.

The lawsuit against RCA was eventually settled in Armstrong's favor, but the victory came a bit late. His wife became very wealthy, but the inventor had already committed suicide in 1954 by jumping from his 13th floor New York apartment. Thus, 2014 is also the 60th anniversary of Armstrong's final (one might say) undertaking. **NV**

"Put the Spark back into *your* electronics"
Solid State Tesla Coils
Plasma Speakers
LED Candle Kits
High Power LED Kits
Eastern
VOLTAGE RESEARCH
Create 4-5 Foot Arcs and music with the Plasmasonic® Musical Tesla Coil
MIDI Compatible
RGB Mood Light Kits
Advanced Strobe Kit (2) 3 Watt CREE LEDs
EXTREME Joule Thief 3 Watt CREE LED
SSTC 1.0 Kit Solid State Tesla Coil $99.99
High Power LED Flame CR2450 Battery
LED Sequencer Kits 24 MODES!
Human Conductivity Kit Body Light 1.0
USB Battery Charger For CR2450 Batteries
Commercial Grade
Designed and Built in the USA
EasternVoltageResearch.com

BY JON MCPHALEN

More Shifty Business

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/march2014_SpinZone**.

Last time we met, we enlisted the aid of a couple old stand-bys — the 74x165 and 74x595 shift registers — to expand I/O in our projects. We even managed to craft some PASM code to provide PWM on multiple x595 outputs. Just before that issue went to print, one of my friends in the Propeller forums speculated about expanding quadrature encoder inputs using the 74x165. Excellent idea, Duane! Thank you very much!

I've never claimed to be the originator of a lot of good ideas. That said, I know them when I see them, and will liberate them when I can (*if it's moral, legal, ethical, etc.*). This is why I try to spend at least a few minutes in the Parallax Propeller forums each morning while I'm having coffee. There are a lot of very smart, very creative people that contribute to the forums, and they generously share a lot of excellent ideas that I take advantage of.

Duane's idea made sense, so I decided I'd give it a whack. I tend to build projects incrementally, so I thought I'd start by converting the ***jm_x165_ez*** project code to PASM. The new file is called ***jm_x165_fast***, and like its predecessor can read from one to four 74x165 shift registers.

The ***start()*** method for the new object has the same interface as the old. Hence, we can go back and forth between the two objects without changing the calling code:

```
pub start(dpin, cpin, lpin)

  stop

  cmd.byte[0]  := dpin
  cmd.byte[1]  := cpin
  cmd.byte[2]  := lpin

  cog := cognew(@entry, @cmd) + 1

  return cog
```

We can pass the pins used in a single long by packing them into *cmd* as shown. I like using the **.byte** modifier in this manner; it takes a few more keystrokes, but is easier for beginners to "see" than:

```
  cmd := (lpin << 16) | (cpin << 8) | dpin
```

With the pins packed into *cmd*, we launch the cog passing the address of *cmd* in the **par** register.

The first step in the PASM code is to unpack the pins, create masks for them, and set as required:

```
dat
                org      0

entry           rdlong   r1, par

                mov      r2, r1
                and      r2, #$3F
                mov      dmask, #1
                shl      dmask, r2

                mov      r2, r1
                shr      r2, #8
                and      r2, #$3F
                mov      cmask, #1
                shl      cmask, r2
                andn     outa, cmask
                or       dira, cmask
                mov     r2, r1
                shr     r2, #16
                and     r2, #$3F
                mov     lmask, #1
                shl     lmask, r2
                or      outa, lmask
                or      dira, lmask
```

Since ***par*** contains the address of *cmd*, we can use it with **rdlong** to get the packed pins — this value is read into cog variable *r1*. To extract the data pin which we've packed into byte0 of *cmd*, we copy *r1* to *r2*, then strip off the other bits by ANDing with $3F. At this point, *r2* holds the data pin. This is converted to a pin mask by moving one into *dmask*, then shifting that left by the value in *r2*. You'll remember that when a cog is started, all pins are set to input mode. Hence, no further action is required.

This process is repeated for the clock and load pins with a couple additional steps. After copying *r1* into *r2*, we shift *r2* right by the appropriate number (eight for the clock pin, 16 for the load pin) to move the pin value to the LSB bits of *r2*. The other bits are cleared and the mask is built. The mask for the clock is used to set that pin to output and low; the mask for the load is used to set that pin to output and high (shift mode of x165).

The next step is to clear *cmd* in the hub as this will be used as the trigger to read from the 74x165 chain. From there, we drop into a loop that waits for a non-zero (eight, 16, 24, or 32) bit count in *cmd*:

```
clear_cmd         mov     r1, #0
                  wrlong  r1, par

get_cmd           rdlong  r1, par      wz
          if_z    jmp     #get_cmd

rd165             andn    outa, lmask
                  nop
                  nop
                  or      outa, lmask
                  nop
                  mov     x165, #0

:loop             test    dmask, ina   wc
                  rcl     x165, #1
                  or      outa, cmask
                  nop
                  nop
                  andn    outa, cmask
                  nop
                  djnz    r1, #:loop
```

This code is a direct PASM translation of the Spin code used in the ***read()*** method of ***jm_x165_ez***. The inputs to the x165 are latched by taking the load pin low (using **andn** with *lmask*), then bringing it back high (using **or**). You'll notice that I extend the pulse a bit with a couple **nop** instructions. I did this because I found that using a solderless breadboard adds capacitance to the system and can have a detrimental effect on high speed signals. You may find these **nop** instructions can be removed with a circuit on a proper printed circuit board (PCB).

After clearing the result variable (*x165*), we drop into a loop that samples the data pin, moving the state of that pin into the C flag (one for high, zero for low). By using **rcl**, the bits in x165 are shifted left and the value of the C flag is placed in bit0 of *x165*. The clock pin is taken high, then back low to get the next bit. The process is repeated until *r1* (desired bit count) is decremented to zero.

The final step is to write the value of *x165* back to the hub, and then clear *cmd* so that the ***read()*** method knows we're done:

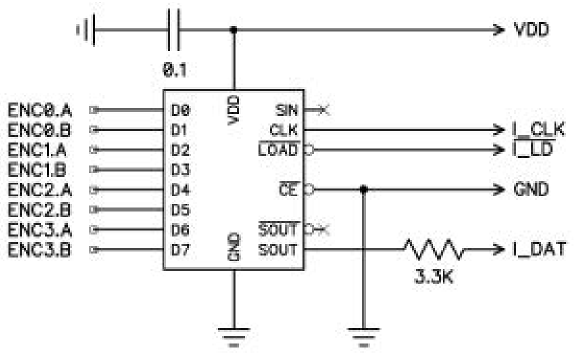
```
                mov      hub, par
                add      hub, #4
                wrlong   x165, hub

                jmp      #clear_cmd
```

There is an object variable called *bits* which is defined just after *cmd* — this means that it will be the next long in the hub RAM space. We copy the address of *cmd* (which was passed in **par**) into *hub* and then add four to point to *bits.* This is used with **wrlong** to copy ***x165*** from the cog to the hub. Finally, we signal the process complete by clearing *cmd* in the hub.

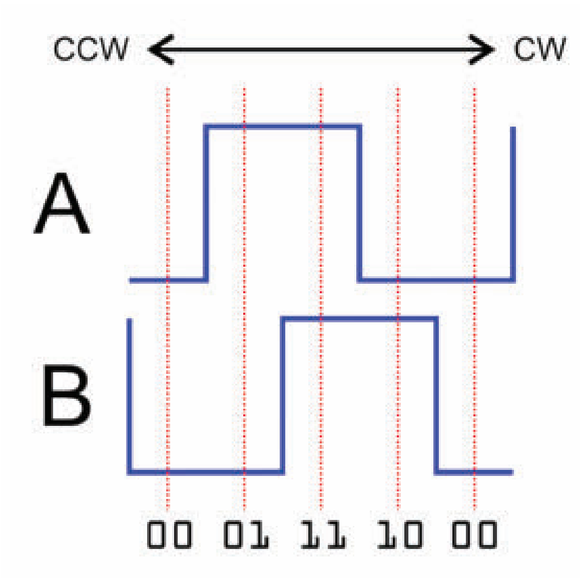
Here is the updated ***read()*** method that interacts with the PASM code:

```
pub read(mode, count)

  count <<= 3

  cmd := count
  repeat
  while (cmd > 0)

  if (mode == LSBFIRST)
    bits ><= count

  return bits
```

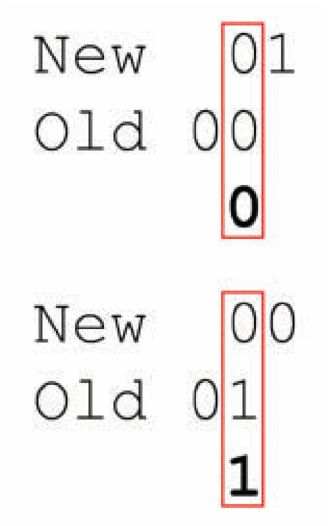
The second parameter is the byte count (1 to 4) which is converted to bits by multiplying by eight (using << for speed). This value is written into *cmd* which will trigger the PASM code. A **repeat** loop monitors *cmd* for the completion of the process. As before, if LSBFIRST



■ FIGURE 1. Encoder to 74x165 connections.



■ FIGURE 2. Encoder phases.



■ FIGURE 3. Direction detection.

mode was specified, we reverse the result bits.

Dialing Up Encoder Inputs

I was toying with the use of a few encoders when I came across Duane's suggestion of using the 74x165 to minimize I/O pins for encoders on a robot. "Great idea!" I thought. Using a single 74x165 would allow me to read up to four quadrature encoders using just three I/O pins.

To do this efficiently, we'll need to establish a rule: The encoder outputs (A and B) will be connected to the 74x165 in pairs, with the A input in the LSB position as shown in **Figure 1**. By using the 3.3K resistor in the data line to the Propeller, we can power the '165 with 5V or 3.3V — selection of Vdd may be affected by the encoders used.

For review, the quadrature encoder has four output phases (**Figure 2**). Note that from phase to phase, only one bit changes (gray code). Detecting movement is easy: The newest reading will differ from the last. When a change is detected, there's a nifty trick to determine direction. The new reading is XORed with the left-shifted old reading; bit 1 (B bit) of the result will hold the direction (zero for CW, one for CCW). You can see this played out in **Figure 3**.

Okay, then. Let's build an encoder reader. Have a look at the ***start()*** method:

```
pub start(dpin, cpin, lpin, x4, p_limits)

  stop

  enc0.byte[0] := dpin
  enc0.byte[1] := cpin
  enc0.byte[2] := lpin
  enc0.byte[3] := x4

  enc1 := p_limits

  x4scale := x4

  cog := cognew(@entry, @enc0) + 1

  return cog
```

Note that we have two more parameters: *x4* and *p_limits*. The first is a four-bit value (%0000 to %1111) that enables the use of detented encoders that output four phases between "clicks."

There are encoders that spin freely; this type is great for wheel tracking on robots. There are others that are better for the human touch; these have detents that allow precision with small changes.

I have a detented encoder that outputs one phase per click (**Figure 4**). I have others that output four phases per click — this is where the *x4* parameter comes in. When using this type, we put a one in the respective bit position of the x4; this will cause the object to divide values coming out of the encoder by four so that we get a change of one per click. Conversely, any values we send into the encoder driver are multiplied by four to ensure proper setup.

The *p_limits* parameter is a pointer to a **DAT** table (of eight longs) that holds the low and high limits of each of the encoders. Using longs we can, technically, have values from negative 2.1 billion to positive 2.1 billion — with single-phase encoders. Even with the x4 type encoders, we can go from negative half a billion to positive half a billion. Yeah, I'm thinking that's plenty of range.

The cog is started and passes the address of *enc0* in **par**; this provides easy access to the packed pins and *x4* bits (in *enc0*), and a pointer to the limits table (in *enc1*).

The unpacking and setup of the 74x165 I/O pins is the same as before. In this object, we're using byte3 of *enc0* to hold the *x4* bits — extraction is a simple matter of making a copy and shifting left by 24. At this point, *r2* contains the *x4* bits:

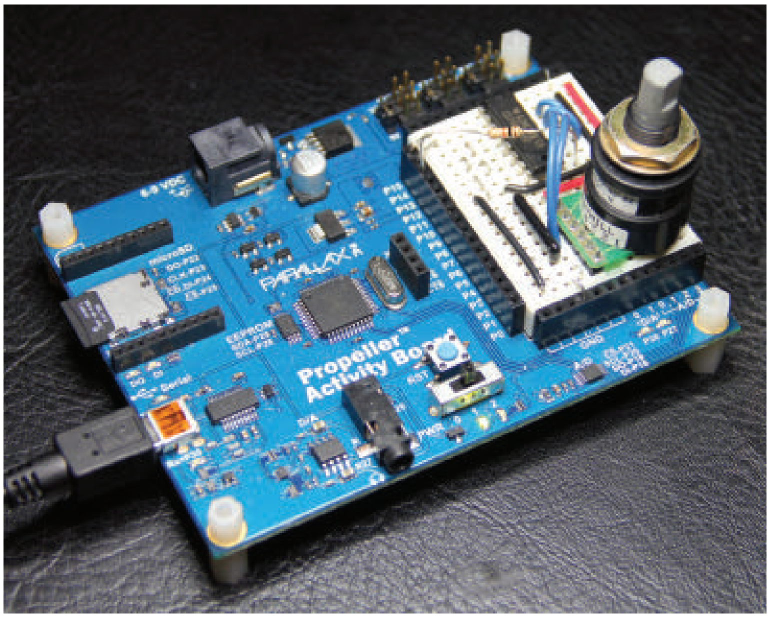
```
              mov      r2, r1
              shr      r2, #24

              add      hub, #4
              rdlong   hub, hub

              test     r2, #%0001  wc
              rdlong   e0lo, hub
        if_c  shl      e0lo, #2
              add      hub, #4
              rdlong   e0hi, hub
        if_c  shl      e0hi, #2
```

In this object, I have a cog variable called *hub* that serves as a pointer to values in the hub RAM. At the beginning, it is assigned the value in **par** (which points to *enc0*). By adding four, we have the address of *enc1*. The next line looks a little odd — what is happening is that we're reading the address (of the limits table) stored in *enc1* into *hub*; now, *hub* points to the limits table.

By using the **test** instruction with the appropriate bit-mask, we can determine the type of encoder; this is saved in the C flag (one for four phases per click, zero for others). The next step is to read the limits for the encoder. If the C flag is set, then we do a left shift on the limit to multiply by four. This process is repeated for both limits for all four encoders (I'm only showing the first).

Note that by reading the limits in the setup for the cog, they cannot be changed on-the-fly — even though Spin allows us to overwrite values in a **DAT** table (more on this later). I elected to code it this way because reading the limits in the working loop of the cog would cut down on the object's bandwidth — which is already slowed somewhat by using the 74x165 to read the encoders. These are the kinds of choices we're forced to make in real world product development.

We're done with the parameters now, so we need to clear them from hub RAM before starting the main code loop:


■ FIGURE 4. Encoder demo on PAB.

```
                        mov      hub, par
                        mov      r1, #0
                        mov      r2, #4
:loop                   wrlong   r1, hub
                        add      hub, #4
                        djnz     r2, #:loop
```

As you can see, this is very easy using a small loop that is the PASM equivalent of **longfill**. The starting address is set to **par** (points to *enc0*), we put zero (fill value) in *r1*, and the number of longs to write (four) in *r2*.

Now, we get to the meat of the object:

```
                        call     #rd165
                        mov      oldscan, x165

encmain                 call     #rd165
                        mov      newscan, x165
                        cmp      newscan, oldscan
                                  wc, wz
              if_e      jmp      #encmain

                        mov      delta, oldscan
                        shl      delta, #1
                        xor      delta, newscan

                        mov      r1, oldscan
                        mov      r2, newscan
                        xor      r2, r1
```

Encoder motion is detected using a comparison of the current reading to the last; this requires an initial reading of the encoder bits into *oldscan* before dropping into the main loop. At the top of the loop, we call ***rd165*** and then compare the values in *newscan* and *oldscan*. If they are different, then one or more encoders has moved between scans.

The value of *oldscan* is copied into *delta*, shifted left by one, and then XORed with *newscan*. We will use selected bits in *delta*, but only if the respective encoder has moved. We can prep for this by copying *oldscan* into *r1*, *newscan* into *r2*, and then XORing *r2* with *r1*.

The value in *r2* can now be used to determine which encoder(s) moved. For any encoder that moved, the AB pair will have changed by one bit which will cause the XOR of the old and new pairs to be non-zero:

```
check0          mov     hub, par
                test    r2, #%00_00_00_11 wz
        if_z    jmp     #check1

                test    delta, #%00000010 wc
                rdlong  r3, hub
        if_nc   adds    r3, #1
        if_c    subs    r3, #1
                mins    r3, e0lo
                maxs    r3, e0hi
                wrlong  r3, hub
```

For each encoder, we set the value in *hub* as required. The next step is to use **test** with a bit mask for the AB pair for that encoder. Remember that **test** works like **and** but does not modify the destination register (*r2*). If the result of **test** is not zero, the encoder has moved between scans.

On movement, we use the encoder's B-bit position in *delta* to determine which direction the encoder moved; this is saved to the C flag (zero for CW [increment], one for CCW [decrement]).

The final step is to read the current value from the hub (into *r3*), then adjust it using the C flag. Note that we're using **adds** and **subs** which treat the value in *r3* as a signed long the same way Spin treats longs. With the value updated, we use **mins** (signed minimum) and **maxs** (signed maximum) to ensure that the encoder value has not strayed past the preset limits. Finally, we write the updated value back to the hub.

This process is repeated for the other three encoders, then the program code loops back to the top.

Let's have a look at the Spin interface methods, which are — as you'd expect — quite straightforward:

```
pub read(n) | value

  if ((n => 0) and (n =< 3))
    value := enc0[n]
    if (x4scale & (1 << n))
      value ~>= 2
  else
    value := 0

  return value
```

The ***read()*** method returns the selected encoder (0..3) value. After the value is read from the encoder array, we check to see if it's an x4 type; if it is, we do an arithmetic right shift (~>) of the value by two bits. This is a quick divide-by-four which preserves the sign for negative values.

We can also preset an encoder value if desired:

```
pub write(n, value)

  if ((n => 0) and (n =< 3))
    if (x4scale & (1 << n))
      value <<= 2
    enc0[n] := value
```

Note that in both the ***read()*** and ***write()*** methods, we're able to access the encoders as an array — even though we didn't explicitly declare them that way. This is a handy feature of Spin that allows us to access variables by their declared name, or as an array element if we desire. The only caveat is that array-type access works only with contiguous variables of the same type.

If you're not at least lurking in the Propeller forums from time to time, you're missing out on a lot of great stuff. Without too much effort, I turned Duane's great idea that he mentioned in the forums into a nifty bit of code.

Propeller Debugging for BASIC Stamp Users

I was enjoying a couple post-holiday beers with my friend, Rick — a special effects expert who taught himself to use the BASIC Stamp — who has put it to use in some big-budget movies. Remember Doc Ock's tentacles in *Spiderman 2*? Rick helped design the maniacal bits at the pincer ends and controlled them using BASIC Stamps.

Rick is transitioning to the Propeller, but sorely misses the ease of the **DEBUG** statement from PBASIC. He's got no trouble using ***FullDuplexSerial*** in the Propeller. What was galling him this particular evening is the apparent inability to output messages to a terminal from multiple objects in the same Propeller project.

To be candid, I wasn't much help that particular

RESOURCES

Jon "JonnyMac" McPhalen
jon@jonmcphalen.com

Parallax, Inc.
www.parallax.com
Propeller chips and programming tools

BILL OF MATERIALS

74HC165	Mouser 595-SN74HC165N
0.1 µF	Mouser 80-C315C104M5U
3.3K	Mouser 299-3.3K-RC

evening. I had used advanced tricks in my own debugging efforts, but I knew Rick wouldn't want to deal with them at this stage of his Propeller game.

We can start multiple copies of ***FullDuplexSerial***, but each consumes a cog (precious resource), has its own set of transmit and receive buffers, and must use its own set of I/O pins to prevent a conflict. What Rick wanted was a serial object that he could use with a terminal from multiple objects; this would allow him to add debugging messages from anywhere within a multi-object project.

It turns out that such a beast exists: It's called ***SerialMirror***. This is, in fact, a copy of ***FullDuplexSerial*** with an important modification that allows its use in multiple objects on the same set of pins. The key difference between ***FullDuplexSerial*** and ***SerialMirror*** is that the variables (buffers, head, and tail pointers) have been moved from RAM (**VAR** section) to a **DAT** section.

The reason is that when multiple copies of the same object are used in a project, the code and data (**DAT** sections) are shared between all copies, though each copy will have its own set of variables (**VAR** section). In ***SerialMirror***, the pointers and buffers are in a **DAT** section, hence shared between all copies. In fact, we can manipulate values in **DAT** sections the same as we manipulate regular variables. By converting the variables to a **DAT** section, they are shared between all copies of ***SerialMirror***.

The key to using ***SerialMirror*** is that it must be started in the topmost object. For child objects, we only need to declare it — do not use the ***start()*** method in child objects. The declaration provides us access to the methods in ***SerialMirror***.

There you have it: The ability to send debugging information to a terminal from any object in your project. I've included ***SerialMirror*** in my encoder demo files at the article link, so you don't have to look for it (it's in ObEx) and you can see it in action, including a mechanism for disabling output from child objects once the debugging is done.

Until next time, keep spinning and winning with the Propeller! **NV**

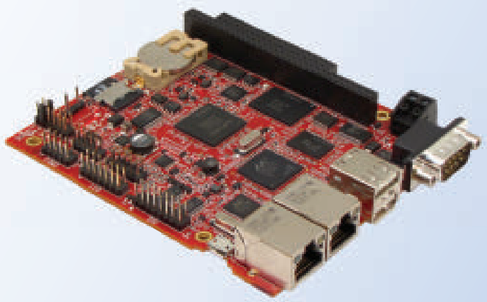
Technologic Systems

We Provide Complete Solutions for Embedded Systems

www.embeddedARM.com
(480) 837-5200

- 30 years in business
- Never discontinued a product
- Open Source Vision
- Engineers on Tech Support
- 45 Day Evaluation Period

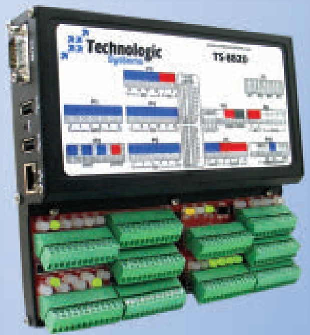
Single Board Computers
TS-7250-V2



pricing starts at
$199 qty 1
$165 qty 100

- Up to 1GHz ARM9
- 512MB RAM
- 4GB eMMC Flash
- Ethernet, USB, CAN, DIO, PC/104
- Low power and sleep mode
- Runs Debian Linux

Industrial Controllers
Rugged, Affordable, Flexible



- Up to 1GHz CPU
- Linux Fast Boot
- Opto-isolated ports
- Screw-down connectors
- Counters & quad decoder
- Debian and GNU tools

Modbus Peripherals

- TS-1400 6 relays
- TS-1420 12 outputs
- TS-1700 DIO, Temp Sensor
- TS-1800 14 A/D, 24 DIO

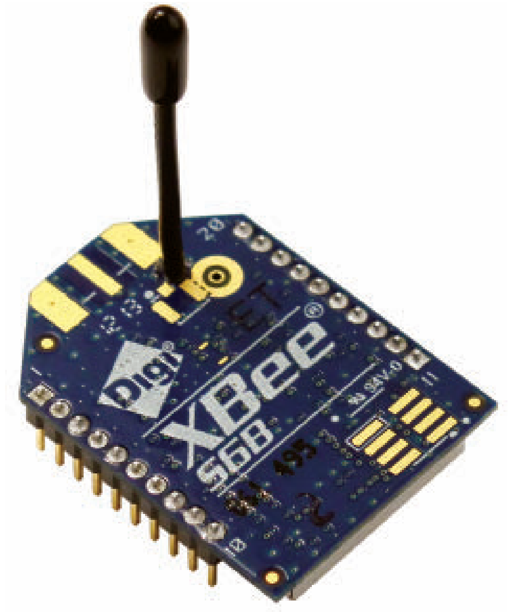
Controllers start at $199 qty 1

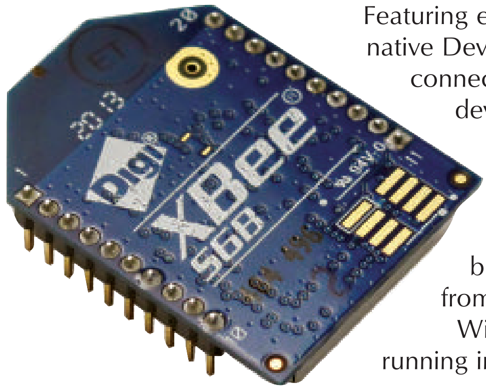
Modbus Peripherals start at $119 qty 100

NEW PRODUCTS

■ HARDWARE
■ SOFTWARE
■ GADGETS
■ TOOLS

XBEE WI-FI WIRE AND PCB ANTENNAS





XBee® Wi-Fi® embedded RF modules provide simple serial to IEEE 802.11b/g/n connectivity. By bridging the low power/low cost requirements of wireless device networking with the proven infrastructure of 802.11, XBee Wi-Fi antennas from Parallax create new wireless opportunities for energy management, process and factory automation, wireless sensor networks, intelligent asset management, and more.

Featuring easy provisioning methods and native Device Cloud by Etherios™ connectivity, XBee Wi-Fi modules give developers the fastest IP-to-device and device-to-Cloud capability possible. Using the 802.11 feature set, these modules are interoperable with other 802.11 bgn devices, including devices from other vendors.

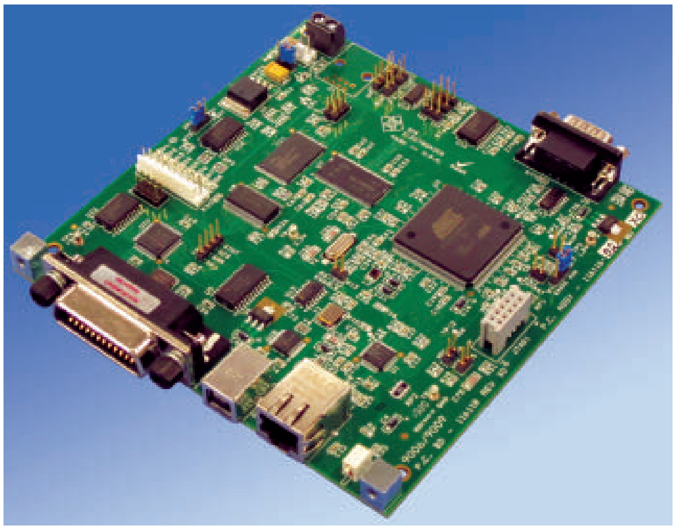
With XBee, users can have their 802.11 bgn network up and running in a matter of minutes.

Key features include:

- Flexible SPI and UART interfaces that provide flexible connection options.
- Supports 802.11 b, g, and n standards.
- 802.11n provides up to a 72 Mbps data rate.
- Price: $36.99.

For more information, contact:
Parallax
Web: **www.parallax.com**

MULTI-FUNCTION MODBUS RTU INTERFACE BOARD



ICS Electronics has announced a new multi-function Modbus RTU interface board for controlling RS-232 and RS-485 Modbus RTU devices. Called the Model 9009, this new interface board allows temperature chambers and other Modbus controlled slave devices to be managed from an Ethernet, GPIB, or USB source or test system.

The 9009's Ethernet port supports VXI-11 and raw socket protocol, Modbus TCP/IP conversion, and an internal HTML web server. VXI-11 and raw socket commands can be used to control the 9009 or any connected Modbus slave devices. Modbus TCP/IP packets are directly converted into Modbus RTU packets.

The 9009's Webserver includes a control page that lets the user enter 9009 or device commands from any Web browser. The 9009 does all of the Modbus RTU packet formatting and handles the response packets. Device responses, Modbus status, and error conditions are displayed for the user on the control page. Switching between the different protocols is automatic and does not require any effort on the user's part.

The 9009's GPIB interface is IEEE-488.2 compliant. GPIB command strings can be used to control the 9009 or be converted into Modbus RTU packets to control any connected Modbus devices.

The 9009's USB interface utilizes Microsoft's universal serial driver that operates on any Windows PC. Using the Microsoft driver avoids future driver issues as Microsoft releases new versions of Windows.

The 9009 has both RS-232 and RS-485 interfaces so it can be connected directly to a single Modbus slave device or it can be connected to multiple Modbus devices via an RS-485 network.

The 9009 is a small 5.5 inch square board that is designed to be mounted on the rear panel of the host chassis with the connectors protruding through the rear panel. The 9009 is designed so the connector shells will make an RFI tight seal with the rear panel.

Pricing for the Model 9009 is $495 each in quantities of one to four units.

For more information, contact:
ICS Electronics
www.icselect.com

USB DATA ACQUISITION **DEVICE**

The USB-16008FS-Plus multifunction USB data acquisition device from Measurement Computing is now available for $399.

Features include:
- Eight single-ended analog inputs.
- 16-bit resolution.
- Simultaneous sampling (one A/D converter per input).
- Up to 400 kS/s sample rate (100 kS/s max for any channel); up to 800 kS/s burst mode.
- Eight digital I/O.
- One event counter.
- External digital trigger input.
- No external power required.
- Out-of-the-box DAQami software to acquire, view, and log data.

Maker Faire®

MAY 17&18, 2014

9th Annual
BAY AREA, CA
SAN MATEO EVENT CENTER
SAT 10AM-8PM, SUN 10AM-6PM

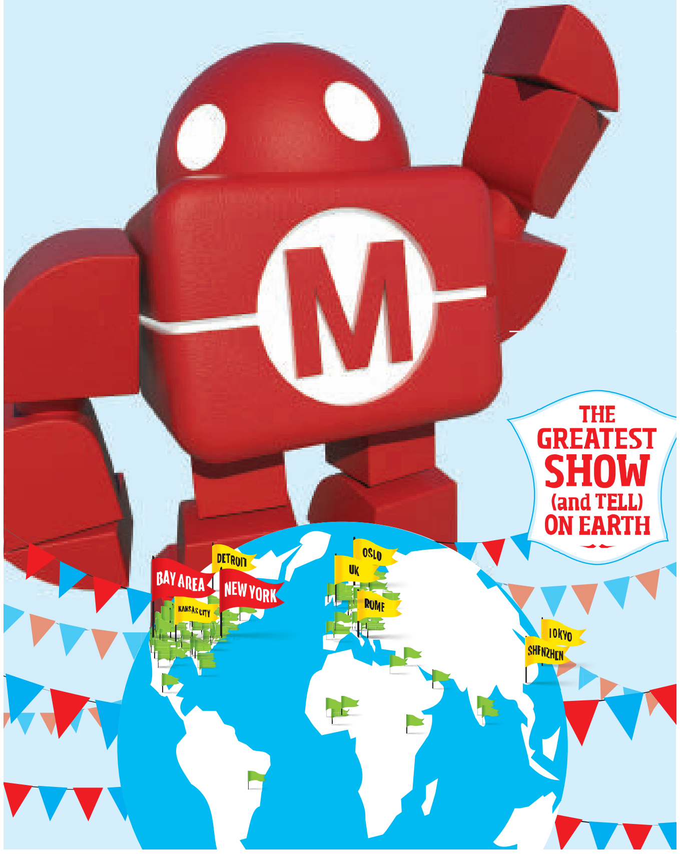
★ ★ ★

makerfaire.com
Brought to you by MAKE magazine

★ ★ ★

Find a Mini Maker Faire near you!
makerfaire.com/map

5th Annual **NEW YORK**
SEPT **20&21**, 2014



TracerDAQ software is included for acquiring and displaying data, and generating signals. A Universal Library includes support for Visual Studio and Visual Studio .NET, including examples for Visual C++, Visual C#, Visual Basic, and Visual Basic .NET.

There's an InstaCal software utility for installation, calibration, and testing, plus ULx for NI LabVIEW, a DAQFlex open-source software framework, and comprehensive drivers for DASYLab.

Supported operating systems include: Windows 8/7/Vista/XP SP2, 32-bit or 64-bit; Linux, Android, and Mac platforms supported by the DAQFlex framework.

For more information, contact:
Measurement Computing
Web: **www.mccdaq.com**

MINI HANDHELD UV-AB METER

Anaheim Scientific has introduced the fifth model in its M-Series of mini handheld environmental meters: the M150 — mini UV-AB meter.

The M-Series are small handheld meters that are designed to make taking measurements out in the field easy and convenient.

Features of the M150 include:
- Wavelength: 290-390 nm.
- Range: 3,999 µW/cm2 and 39.99 W/cm2.
- Accuracy: ±4% + two digits in sunlight (15% for other light sources).
- Price: $129.

Other M-Series meters already released include a mini light meter, a mini temperature/humidity meter, a mini anemometer, and a mini solar power meter.

The M-Series meters have the following features in common:
- Large 3-3/4 digit display.
- Small 5.3 x 1.9 x 0.2 inch size weighing only 2.8 oz (80 grams).
- Data hold.
- Maximum/average/minimum hold.
- Zero adjustment.
- Low battery indicator.
- Auto power off and disable function.
- Light weight (less than 9 oz).
- Powered by two AAA batteries.
- Limited two year warranty.

For more information, contact:
Anaheim Scientific
Web: **www.anaheimscientific.com**

RBC LOW VOLTAGE CONTROL AND POWER INTERCONNECT

The J2 LED Lighting, LLC Red Black Pair Cable (RBC) is a UL2464 bulk electronic grade black jacketed flexible interconnect cable with 18 AWG red/black conductors for DC power applications. This is a UL2464 rated indoor use cable that is flexible and of a small 5 mm outer diameter. It has a smooth round pressure extruded outer jacket to remove convolutions, and also has good flexibility with a 5/8 inch typical bend radius. The round cable coils well for storage when used for portable power applications.

The UL2464 two conductor red/black pair 18 AWG cable type is suitable for low voltage control and power interconnect applications for LED driver power supplies, dimmers, motion controls, and LED modules. The UL2464 specification covers the use of cable for internal wiring or external interconnection of electronic equipment. The cable carries the VW-1 marking for flame retardancy testing covered under UL1581.

Specifications include:
- AWG: 18 (two PVC insulated conductors red and black) bare copper flexible 41 strand.
- Amp: 10 max.
- Volts: 300 max.
- System volts: 12–24 typical.
- Flammability: VW-1.
- Certifications: UL component (cable is marked UL).

Recycling & Remarketing High Technology

WEIRDSTUFF® WAREHOUSE

Software, Computers, Electronics, Equipment, Doohickies

WE BUY/SELL EXCESS & OBSOLETE INVENTORIES

FREE COMPUTER AND ELECTRONIC RECYCLING

GIANT 10,000 SQ. FT. AS-IS SECTION

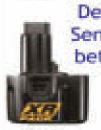
384 W. Caribbean Dr.
Sunnyvale, CA 94089
Mon-Sat: 9:30-6:00 Sun: 11:00-5:00
(408)743-5650 Store x324
WWW.WEIRDSTUFF.COM

---

www.Primecell.com

Battery rebuilding service

Dead Batteries ? Don't toss them. Send them to us - our rebuilds are better than original specifications.



Tools: Hilti, Skil, Milwaukee, Panasonic, B&D DeWalt, Makita, All 2-36 Volts

Electronics: Bar Code, Scanners, Surveying, Printers, Laptops, Photography

Radios: APELCO, UNIDEN, G.E., ICOM, KENWOOD, MOTOROLA, MIDLAND, MAXON, YAESU, ALINCO

Visit www.primecell.com for important details
24 Hr Secure recorder tel-fax (814) 623 7000
Quotes email: info@primecell.com
Cunard Assoc. Inc. 9343 US RT 220 Bedford PA 15522

LEMOS INTERNATIONAL

ZigBee Home Automation

ZigBee Pro

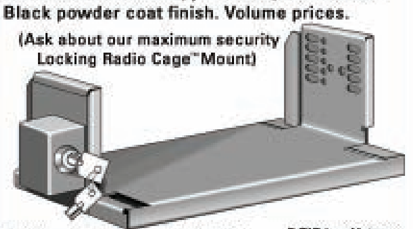
ZigBee Range Finder

www.lemosint.com

SLIDE MOUNT for MOBILE RADIOS
UNIVERSAL: FITS MOST 2 WAYS

Model ET-850 Universal Slide Mount. Standard mounting platform, specify radio make/model. Ends blind fumbling to release hidden fasteners. Easily transfer radios from vehicle to vehicle. Also use in the office, job trailer, marine too. Black powder coat finish. Volume prices.

(Ask about our maximum security Locking Radio Cage™ Mount)



E TIP, Inc., North Aurora, IL 60542 USA
(800) 530-5064 FAX (630) 801-9569
www.etipinc.com | sales@etipinc.com

E TIP is a Veteran Owned Small Business

Quality Kits for All Your Electronic Needs!

LOTS OF GREAT GIFT IDEAS!

Explore the World of Electronic Project Kits and Devices.

We offer a wide selection of electronic kits for beginners, as well as more advanced projects for students and experts! 1-888-GO-4-KITS

We also carry Arduino and Raspberry Pi products, as well as a variety of speed controllers, timers, power supplies, plus lots more!

QKITS.COM

NKC electronics

MDE8051 Trainer by Digilent

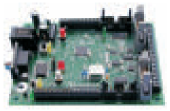
NKCelectronics.com/MDE8051

Includes the MDE8051 training board, power supply, serial cable

Purchase Orders are accepted from Educational Institutions, US Government and Research Centers

---

ROBOT POWER *Extreme Motor Speed Control!*



- Closed-loop control of two axes
- **Windows GUI for all features**
- Full PID position/velocity loop
- Trapezoidal path generator
- Giant Servo Mode!
- *www.embeddedelectronics.net*

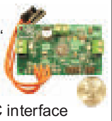
Dalf - **$250**

MADE IN THE USA

1.25"x .5" **$29.99**

Bots IQ Favorite

1.6" x 2.7" **$119.99** *Powerful!*



Scorpion Mini - Single **6A** H-bridge - R/C interface
Scorpion XL/XXL - Dual 13A/25A H-bridge - R/C
>> Also H-bridges, gearmotors, plans, schematics <<

www.robotpower.com
Phone: 253-843-2504 • sales@robotpower.com

Redwood Electronics Corporation

industry-specific closed circuit video equipment

------

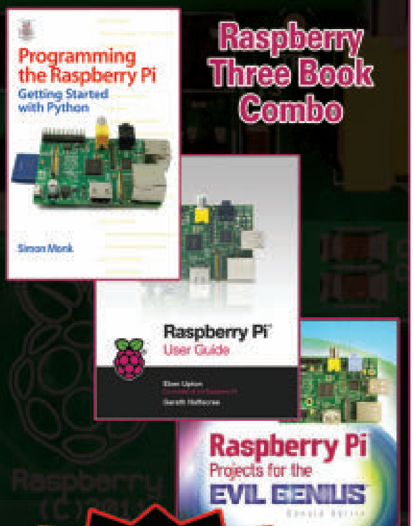
service-oriented consumer parts department

call: (800) 227-1768

In the Nuts & Volts Webstore NOW!

NUTS AND VOLTS

NV BOOK SPECIALS



Raspberry Pi Three Book Combo

Only $48.95
Plus
FREE Priority Mail Shipping
US Only

To order call 800 783-4624 or visit:
http://store.nutsvolts.com

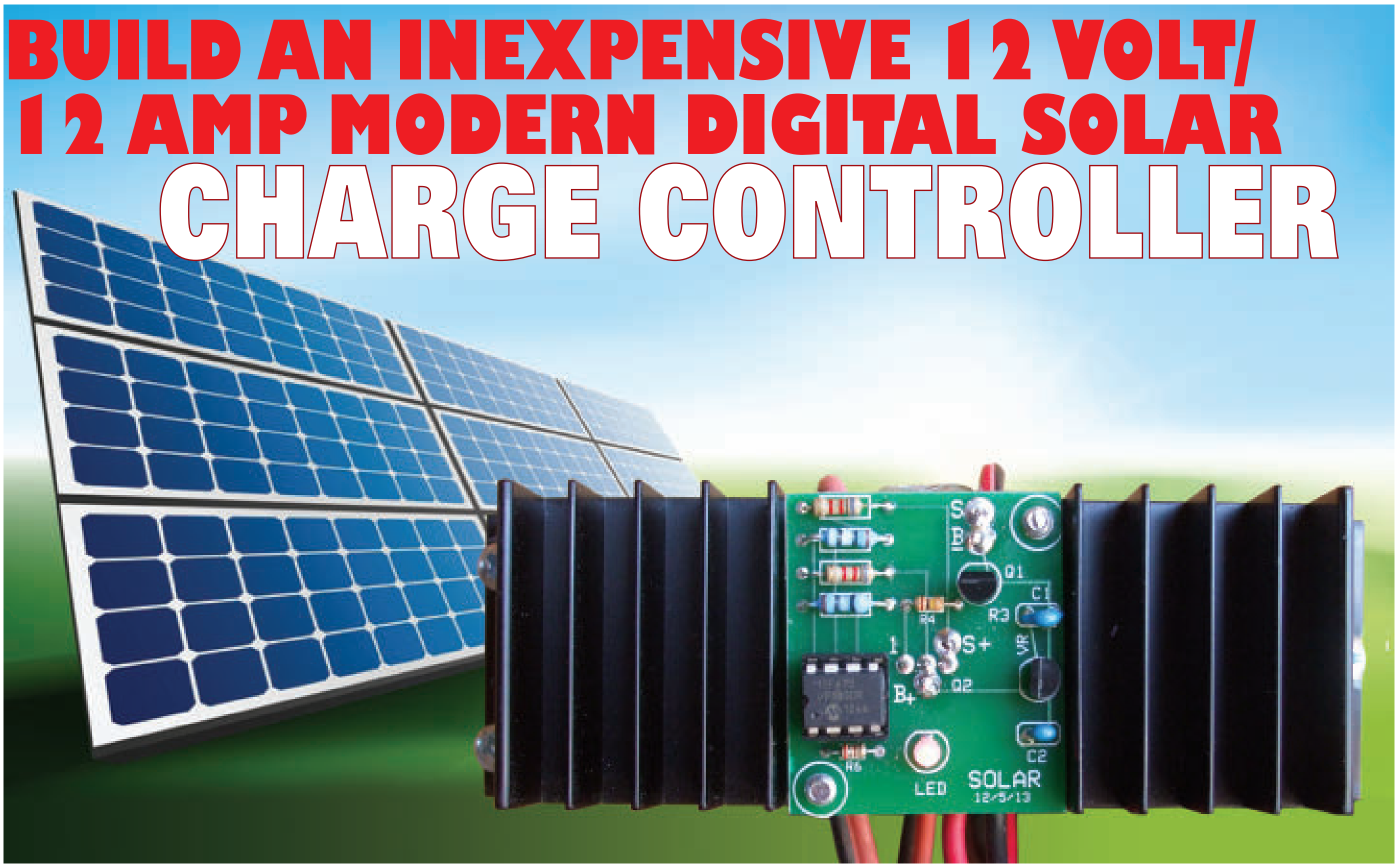
Limited time offer

- Maximum operating temperature: 80°C dry
- Composition: Rohs/Reach
- OD of Jacket Insulation: 5.0 mm (0.197") typical.
- OD of inner conductor insulation: 1.85 mm (0.073") typical.

The J2 LED Lighting RBC-18AWG-UL2464 is jacketed with abrasion resistant PVC; this uniformly round outer jacket provides a good seal with standard cable grips. A proper size fuse or circuit breaker is recommended to protect the cable circuit from overload currents. Circuit protection should be set to a maximum of 10 amps. For applications with the cable bundled with other cables or with operating ambient temperatures above 50°C (122°F), the circuit protection should be lowered to 7.5 amps maximum. The RBC-18AWG-UL2464 is suitable for many applications including the following:

Continued on page 77

BUILD AN INEXPENSIVE 12 VOLT/ 12 AMP MODERN DIGITAL SOLAR CHARGE CONTROLLER



By Ron Newton
sjnewt@att.net

Go to **www.nutsvolts.com/index.php?/magazine/article/march2014_Newton** for any additional files and/or downloads associated with this article, or to discuss this topic.

Several readers have asked the *Nuts & Volts* staff to come up with an inexpensive solar controller kit for charging 12 volt lead-acid batteries. So, I decided to tackle the project. One of the first things I do when I design something is ask for specifications of what is needed.

The specifications for this project (given by one of the readers) were to control the charging of lead-acid batteries up to 12 amps using solar cells. He wanted the unit to start charging if the battery voltage went below 13.5 volts, and then turn the charging off when it reached 14.25 volts.

This project is very inexpensive and fits onto a heatsink that is mountable. Since the voltage is low, there is no danger of shock. It is simple to build, ideal for the novice, and no special tools are needed other than a soldering iron and a 9/64" drill.

Battery Functions

To charge a battery, the charging voltage of the charging device — whether solar or a regular battery charger — has to produce a voltage above the battery voltage itself. If you think of a battery as a big resistor, the picture becomes clearer. Lead-acid batteries have a resistance of approximately 20 milliohms. However, the charging resistance of a battery appears to be considerably higher. The amperage will also be limited by the output of the charging device and its resistance. It will merrily drop its charging voltage which will limit its charging amperes.

If the resistance of the charger is one ohm, it produces 14.5 volts; the battery is at 13.5 volts, so its output is limited to:

1 *volt* / 1 *ohm* = 1 *amp*

If one connects a bank of solar cells to produce voltages over 14.5 volts, the solar panel will continue to overcharge the battery which will cause the battery acid to disassociate its water into oxygen and hydrogen, and the battery will get hot and boil:

$2\ H_2O = 2H_{2\uparrow} + O_{2\uparrow}$

Therefore, the charging voltage should be limited to 14.25 volts. I was concerned this might be too high, so I lowered it on this project to 14.00 volts. Gassing will take place at 14.25 volts if the temperature exceeds 30°C. On the other hand, if the battery is allowed to discharge completely, it will form lead sulfate crystals. These are hard to get back into a solution. This is called sulfation:

$Pb(solid) + PbO_2(solid) + 2H2SO_4(aqueous) \rightarrow 2PbSO4(solid) + 2H_2O(liquid)$

Lead-acid batteries are constantly discharging due to their nature. They will lose approximately 1% of their charge per day. This is why you should never store lead-acid batteries for long periods of time. In the good old days, the rule of thumb was to use a trickle charger, and it is still a good idea. It was often a small transformer with a diode that provided one half wave rectification that produced 13.5 volts at .25 amps.

Older voltage regulators in cars use a simple regulator with a heavy relay. The coil of the relay was connected across the generator, and its points connected the generator to the battery. When the generator is producing voltage, it induces a magnetic field into the coil and closes the contacts. The contacts allow the battery to charge.

When the generator slows down, there is not enough current to hold the contacts closed, so it disconnects the generator from the battery. As you can see, this system does have the potential to overcharge a battery since nothing is measuring the voltage of the battery to see if it exceeds 14.25 volts.

Some of you are probably asking why we need to disconnect the battery from the generator. When the generator is not generating, it becomes a motor.

Wow! I think I just discovered a potential perpetual

PARTS LIST

ITEM	DESCRIPTION	PART #	QTY	COST	TOTAL	SOURCE
Printed circuit board						Express PCB
C1	0.33 µF	810-FK28X5R1E334K	1	$0.28	$0.28	Mouser
C2	0.1 µF	81-RDEF51H104Z0K103B	1	$0.24	$0.24	Mouser
IC1	Microchip	579-PIC12F675-I/P	1	$1.21	$1.21	Mouser
LED 1	Bi-colored LED	160-1058-ND	1	$0.59	$0.59	Digi-Key
Q1	N-MOSFET 2N7000	119423	1	$0.07	$0.07	Jameco
Q2	P Power MOSFET	IRF5305PBF	1	$1.25	$1.25	Jameco
R1-R3	8200 1%	18.2KXBK-ND	2	$0.10	$0.20	Digi-Key
R2-R4	909 1%	909XBK-ND	2	$0.10	$0.20	Digi-Key
R4	100K	100KEBK-ND	1	$0.10	$0.10	Digi-Key
R5	220 1/6 watt	220EBK-ND	1	$0.10	$0.10	Digi-Key
VR	5 volt 7805	497-2952-5-ND	1	$0.39	$0.39	Digi-Key
Heavy duty heatsink	1^{o} C/W	GHT144-R	1	$3.75	$3.75	Jameco
Spacer	1/4" nylon	492-1104-ND	2	$0.08	$0.16	Digi-Key
Screw	6-32 3/8" nylon	H556-ND	1	$0.09	$0.09	Digi-Key
Nut	6-32 nylon	H620-ND	1	$0.11	$0.11	Digi-Key
Screw	6-32 5/8"	H164-ND	2	$0.03	$0.05	Digi-Key
Socket	8-pin IC socket	AE9986-ND	1	$0.18	$0.18	Digi-Key
Thermal pad	Thermal pad	926-1483-ND	1	$0.11	$0.11	Digi-Key
Wire	Black 18 gauge stranded 1 ft	125787	2	$0.16	$0.32	Jameco
Wire	Red 18 gauge stranded 1 ft	125736	2	$0.16	$0.32	Jameco

motion machine! Why not take a circular raceway (or better yet, an oval raceway) and raise one end. Take a generator and battery, connect them together, and put them on a cart. When the cart goes downhill, it will charge the battery; when the cart starts uphill, the generator will become a motor and drive the cart uphill.

But I digress ... back to the use of relays. Relays are good as they have very low contact resistance. However, the contacts do arc and the setup is mechanical. (Not good for speed nor endurance.)

Silicon devices such as triacs, SCRs, transistors, FETS, etc., have the advantage there is not much to wear out and they can work at high speeds. They will burn out, however, if not properly heatsunk. Where does this heat come from? Silicon devices like diodes and transistors often have a voltage drop of something equal to .8 volts. The FET I'm using is a P-MOS IRF5305PBF which has a resistance of .06 ohms from source to drain when running in the saturated mode. One watt will produce 14.3 calories per minute since a calorie is equal to raising 1 gm (or at 1 ml) of water 1°C in one minute. So, 14.3 calories would raise 14.3 gms of water one degree. A tablespoon of water is about 15 gms.

When 12 amps are drawn with a resistance of .06 ohms, the voltage drop across the FET is .72 volts:

.72 volts * 12 amps = 8.7 watts or ≅ 120 calories

If we take a tablespoon of water at 25°C (room temperature) and place it on the transistor, in one minute it will raise the tablespoon of water to 34°C (93°F). In seven minutes, the water would boil. Now, you should understand why we need to heatsink silicon devices. A great website for the calculation of heat transfer to a heatsink is at **www.mustcalculate.com**.

The Circuit

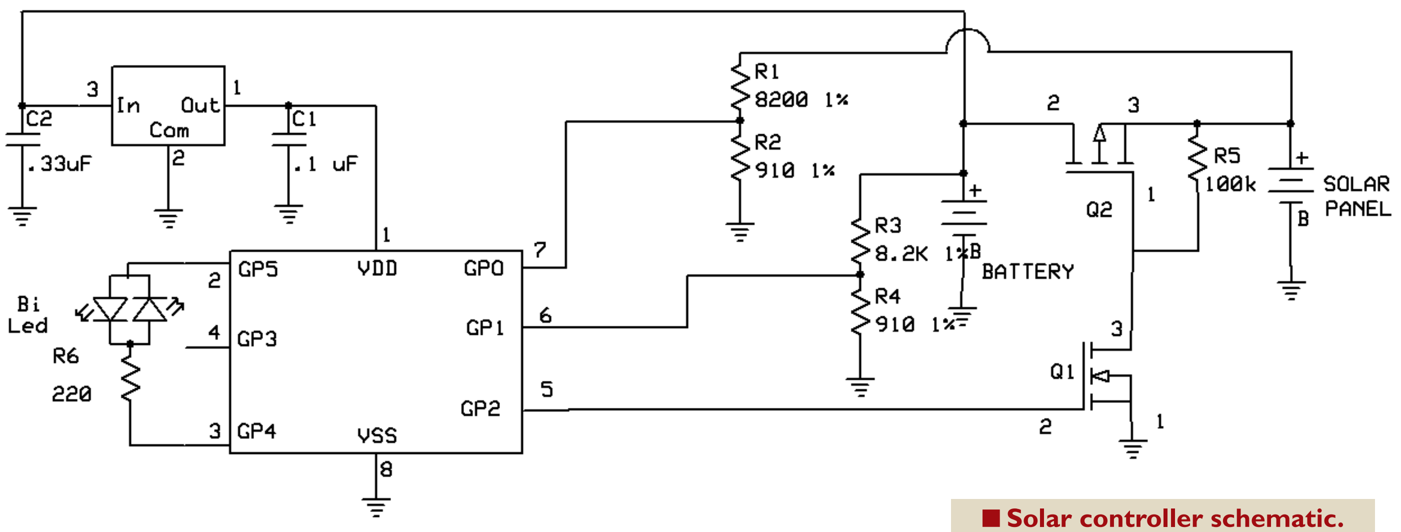
To measure the voltage of the battery, I used a PIC16F675 which has a 10-bit A/D (analog-to-digital) converter. This allows you to get down to /.0049 volts per bit (five volts/1,024 bits). A voltage divider using 1% resistors reduces the voltage down 1:10, so 14.00 volts cut-off voltage will be 1.400 volts, or 286 bits. The turn-on voltage is set to 13.5 volts which equates to 1.35 volts or 275 bits. The PIC is powered at five volts with a 7805L using the battery voltage; see the **schematic**.

A P-MOSFET is used to turn the solar panel on and off. The one used here is capable of sinking 31 amps. Keep in mind, though, the heatsink is the key. MOSFETs have the advantage of a low resistance when they are run in the saturated mode. A second N-MOSFET is used as a voltage translator as the five volts from the PIC will not totally turn off the P-MOSFET.

The P-MOSFET is run either fully on or off. It is connected to the positive lead of the battery from the positive lead of the solar panel. The PIC measures the voltage on the battery by bringing the FET high, thus disconnecting the solar panel from the battery. It then determines if the battery needs to be charged by measuring its voltage.

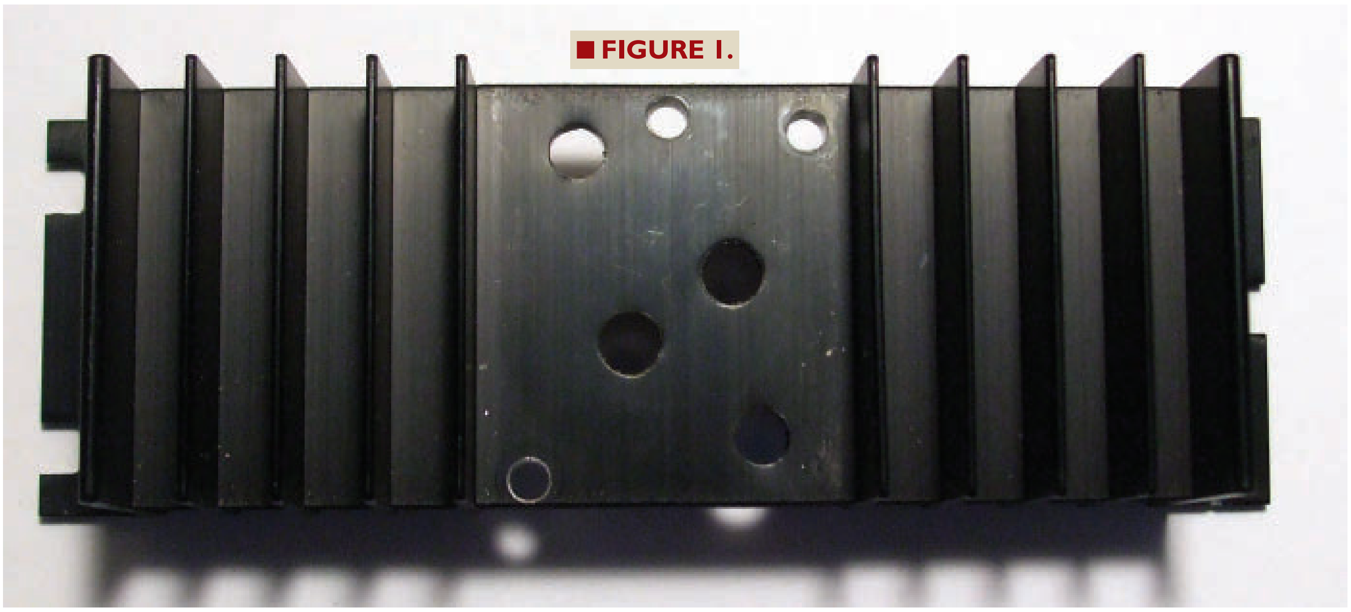
If it does need charging, it connects the positive line of the solar panel to the positive terminal of the battery. If not, it continues to monitor the voltage until the battery voltage goes below 13.5 volts. It then connects the positive side of the solar panel to the positive terminal of the battery and allows its voltage to charge the battery.

To prevent the unit from drawing excessive power, I measure the voltage across Q2 that controls the



■ Solar controller schematic.

■ FIGURE 1.



charging. The PIC measures the battery voltage and the solar panel voltage, and takes the difference. If the voltage is over 1.25 volts (about 20 amps), it goes into the alarm mode and turns the FET off. The LED will blink off and on red. It will automatically reset.

FETs will also conduct both ways, thus creating a potential that the batteries could discharge through the solar panel. To prevent this, an algorithm is used, and when the solar voltage drops below the battery voltage the unit disconnects from the solar panel. This prevents discharge when the sun goes behind a cloud or the sun goes down. Once every 10 minutes, the battery is disconnected from the solar cell and its voltage is checked. When the battery reaches 14.00 volts, the solar panel disconnects and the micro continues to monitor until the battery's voltage reaches 13.5 volts. It then re-connects to the solar panel.

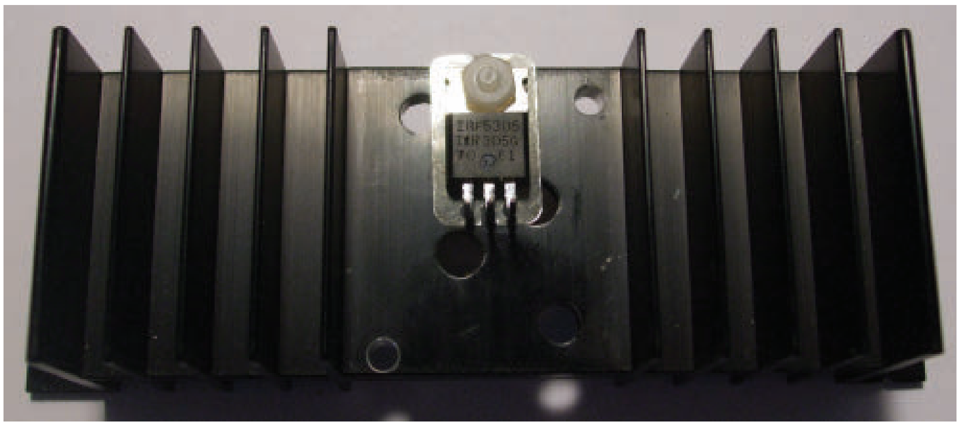
Building the Circuit

A kit is available from the *Nuts & Volts* Webstore. It comes with a preprogrammed chip and board. If you want to program your own chips, you will need a programmer. The printed circuit board (PCB) files are available at the article link, along with the assembly and hex files for the PIC, plus some usage hints and tips, and a copy of the Parts List. The board files are from ExpressPCB and can be downloaded with their free software. If you are a programmer, you can change the tripping voltages if you feel so inclined. (Refer to the Hints and Tips file at the article link.)

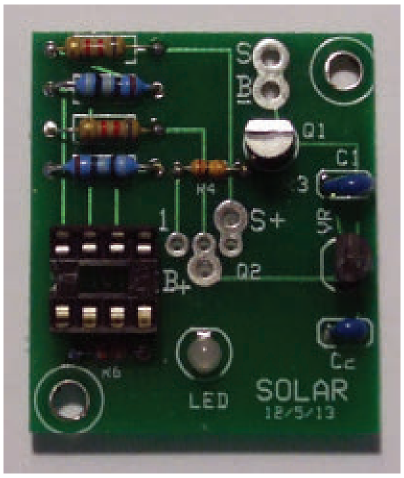
Place the heatsink with its mounting holes facing down; rotate it so that its single mounting hole is to the right. Using the template available with the downloads, cut the template out and glue it on the top of the heatsink. Drill three holes using a 9/64"drill and deburr. Make sure the heatsink is flat with no burrs. Remove the template with hot water. Take a look at **Figure 1**.

With Q2 facing you, bend its leads toward you 90 degrees so they are facing up. To prevent the heatsink from becoming a conductor, locate the insulating pad and place it on the heatsink. Mount the FET using the 6-32 nylon screw and nut. The screw should be on the back of the heatsink; the nut should be on top of the FET. The purpose of this is to allow the removal of the transistor and board without having to de-solder. Refer to **Figure 2**.

■ FIGURE 2.



■ FIGURE 3.



Solder the six resistors and two capacitors to the board. If you have trouble reading color codes, use an ohmmeter. The board is marked. Solder the voltage regulator to the board with its flat pointing to the center of the board and Q1 to its pads noting its markings. Solder the socket for IC1 with lead 1 going to the square pad. Solder the LED with its long lead to the square pad. Check out **Figure 3**.

Solder two #18 black wires to the negative pads and two #18 red wire to the positive pads as in **Figure 4**.

Some of you will be asking, "Wouldn't it be better to move the pads for the wires closer to the edge?" It would be more convenient. However, PCB traces don't fare well with high amperage. If you calculate out the width of the trace needed to transfer 12 amps with 1 oz of copper trace, it would have to be .5" wide — one-third of the board width.

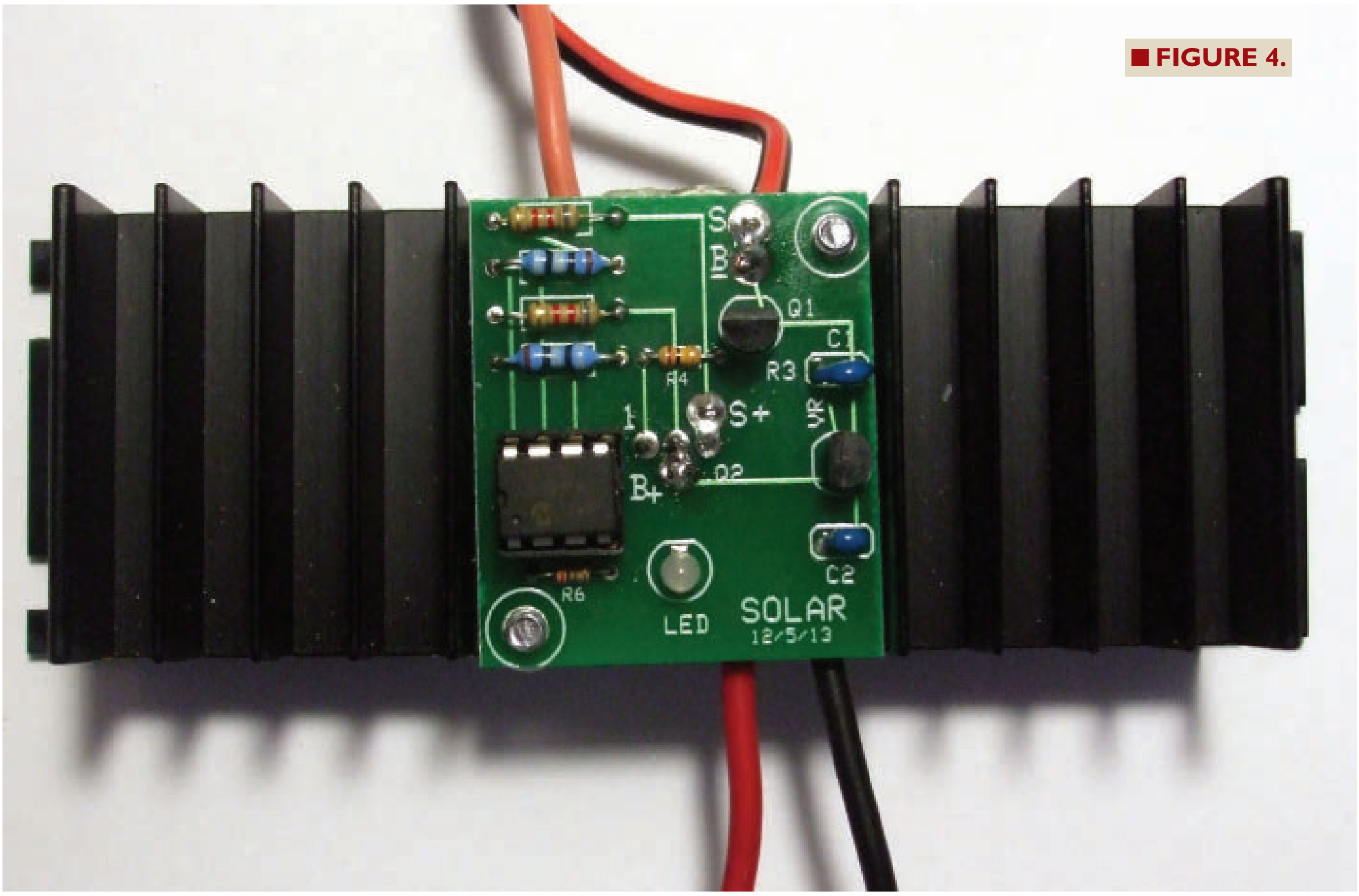
Thread Q2's leads through the board transistor pads. Speaking from experience, make sure the resistors are next to the body of Q2 (**Figure 5**). Mounting the board in backwards will reverse the leads of Q2 and will cause failure. Use the two 1/4" nylon spacers and two 5/8" 6-32 screws. Put the screws through the back of the heatsink and screw them into the board. The board will self-tap. Solder Q2's leads.

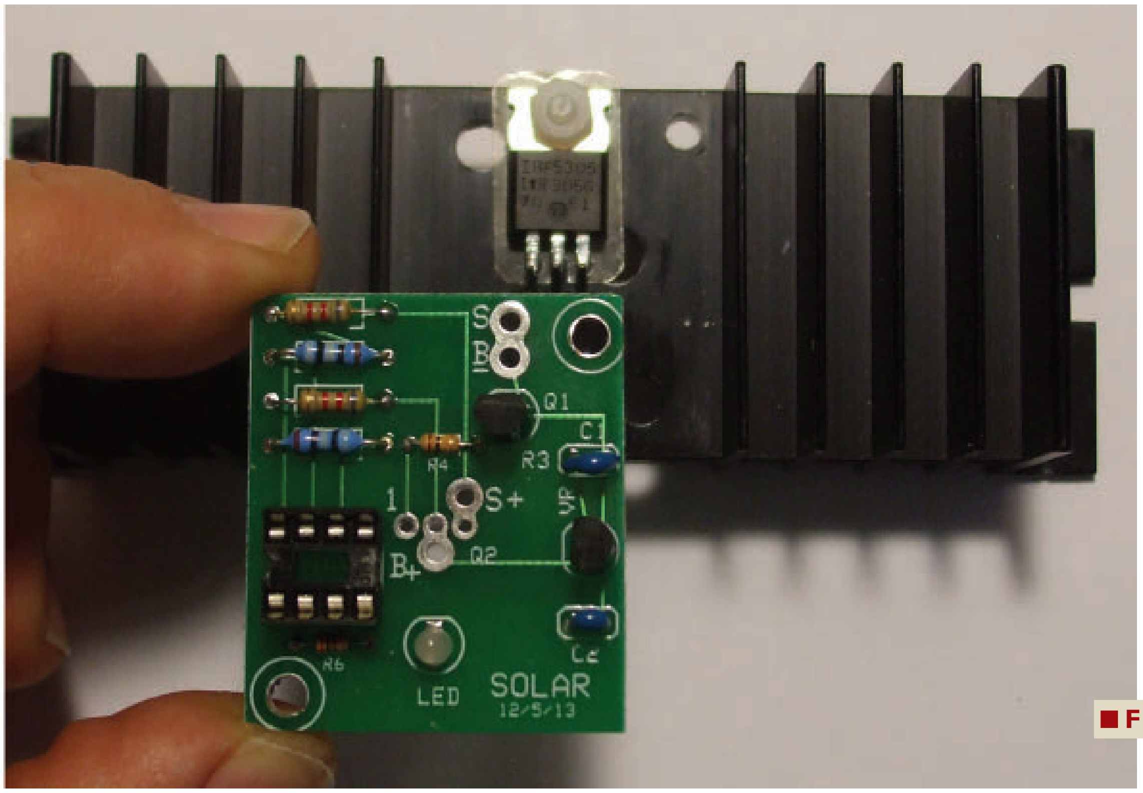
There are mounting slots on the heatsink for mounting the unit.

Using the Unit

Using wire nuts, connect the unit to the batteries first. The LED should turn on. Green shows the batteries

■ FIGURE 4.





■ FIGURE 5.

are charged (or the solar panel voltage is below the battery voltage); red indicates the batteries are charging. Now, connect to the solar panel.

You might want to consider adding an amp meter and an inline fuse. The LEDs only show what the battery is calling for and does not indicate that the solar panel is charging.

Happy are those who keep themselves (and their batteries) charged up! **NV**

Did You Know
Preferred Subscribers get access to all the digital back issues of *Nuts & Volts* for free?

Call for details
1-877-525-2539
or go to
www.nutsvolts.com/faqs

Feedback Motion Control

The Old Way
1) Build robot
2) Guess PID coefficients
3) Test
3a) Express disappointment
3b) Search Internet, modify PID values
3c) Read book, modify PID coefficients again
3d) Decide performance is good enough
3e) Realize it isn't
3f) See if anyone just sells a giant servo
3g) Express disappointment
3h) Re-guess PID coefficients
3i) Switch processor
3j) Dust off old Differential Equations book
3k) Remember why the book was so dusty
3l) Calculate new, wildly different PID coefficients
3m) Invent new, wildly different swear words
3n) Research fuzzy logic
3o) Now it is certainly not working in uncertain ways
3p) Pull hair
3q) Switch controller
3r) Re-guess PID coefficients
3s) Switch programming language
3t) Start a new project that doesn't need feedback control
3u) See parts in box. Feel guilty. Go back to old project
3v) Start testing every possible combination of PID coefficients
3w) Apply eye drops to red, bleary, sleep-deprived eyes
3x) Wait, it's working?
3y) Decide not to do any more projects that require control systems
3z) Wonder why someone doesn't just make a thing that tunes itself

The Kangaroo x2 Way
1) Build robot
2) Press **Autotune**
3) Get a snack

Kangaroo x2 adds self-tuning feedback to SyRen and Sabertooth motor drivers.

$24.99

DimensionEngineering

www.dimensionengineering.com/kangaroo

A MULTI-STAGE HIGH SPEED INPUT MONITOR

Remove the doubt! Monitor those competitive events with this low cost, portable, and easy-to-build project!

By John Mastromoro
jfmcircuits@frontiernet.net

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/march2014_Mastromoro.**

Using baseball as an example, when it comes to sports, umpires are usually the ones in the 'hot seat' — especially when a close call decision is made. Sometimes a video playback is used for confirmation, but usually umpires are pretty good at being accurate. However, it certainly doesn't mean they can't make a questionable call now and then.

The project in this article has proven itself to be a superb "umpire" that will make the right decision — even among 10 inputs at once that are all within a ~100 µS time frame. During testing, I used four phototransistors wired in a single row, one-half inch apart. Each was wired to an identically-configured input, just as indicated in the **Figure 1** schematic.

Using a handheld IR emitter and doing it several times, I swiftly passed it over and across the row of phototransistors as fast as I could; first from left to right with a 1, 2, 3, 4 display; then, from right to left, displaying a 4, 3, 2, 1, and not one failure!

Commercially made high speed event monitors basically operate the same and — as I discovered — are comparatively quite expensive. They are typically used to monitor *final inputs* and display the associated 'input order numbers' anywhere from slot-car racers to swimming meets, as well as being used for *reaction time* events if so desired. Most are constructed with lapse timers indicating start/finish times, but even those *without* this added feature can be quite expensive.

After comparing the electrical specs of commercially available units [~100 µS], I determined that the project described here is equally as worthy, offering *multiplexed* monitoring [only without lapse timers] for one to 10+ inputs, with high speed input accuracy and far less expense than commercial units [although with lapse timers] that monitor only two inputs. Commercial units can be priced from several hundreds of dollars to several thousands of dollars, depending on how many inputs you want to monitor.

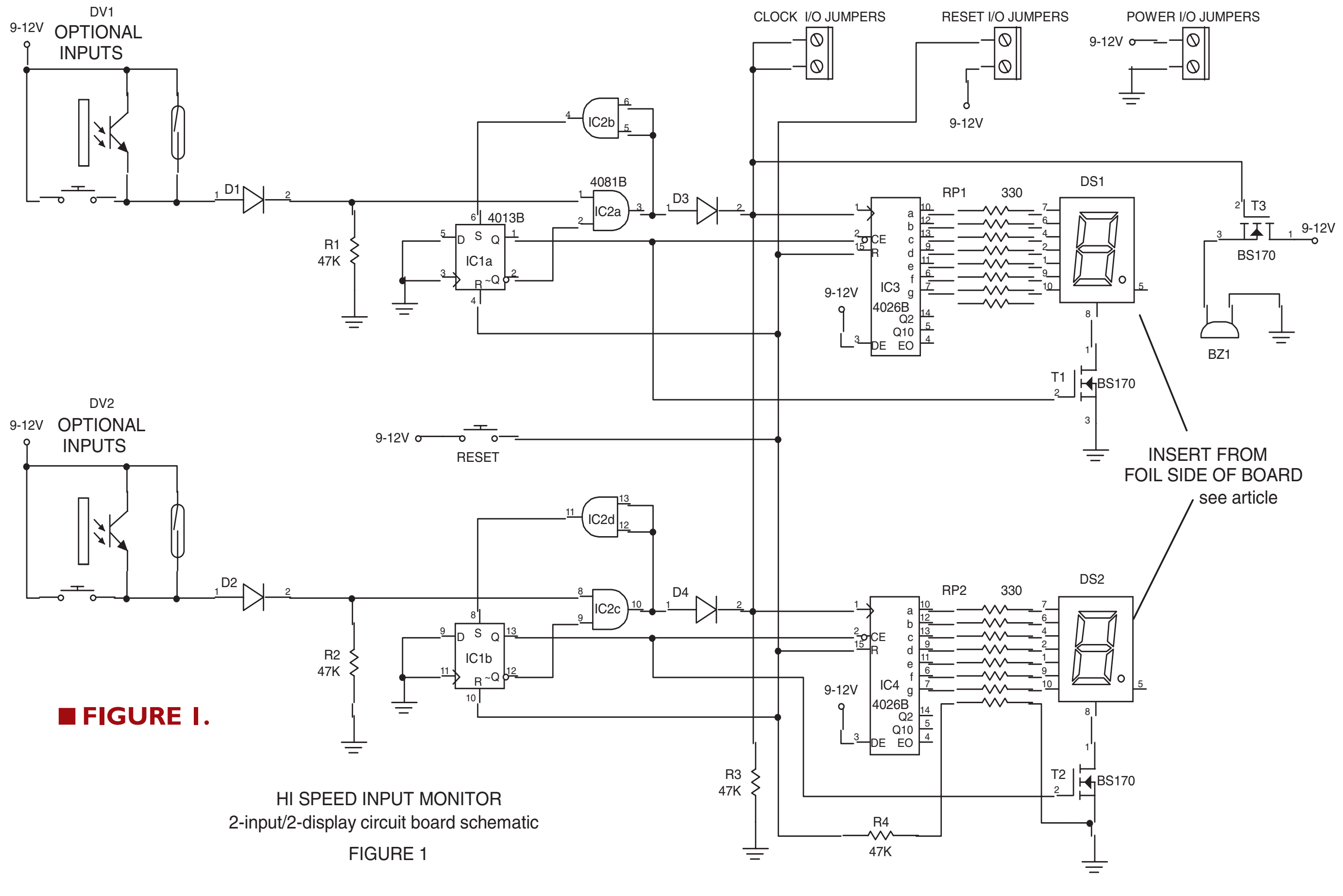
Project Nuts and Bolts

Although this article refers only to the two-input board, I do have a layout for a single four-input board available but it's not shown here to comply with the circuit schematic. However, the four-input board is simply two two-input boards configured as a single board [eliminating a few jumpers]. Anyone that desires the four-input board layout, files, etc., can simply email me and all necessary files will freely be sent.

To construct the two-input monitoring project (with circuit board) will cost approximately $20-$25; with a single four-device input board, it would be approximately $30-$35 excluding the optional input device costs and enclosures used. I do have some two-input boards available, but if requests are sufficient enough to order more [to keep board costs down], I will place accommodating orders.

The timer circuitry is not necessary for this project to perform, therefore is not included since it can be installed as an add-on feature. The idea of this project is to offer an uncomplicated, easy-to-construct inexpensive 'order number' input monitor.

Upgrades will follow this project such as wireless data transmission and MP-controlled 'lane-split' timer data indicating the finalized time differentials between all inputs (eliminating conventional start-to-end timing).



■ FIGURE 1.

To DIY or Not to DIY

I have never been involved with this sort of project before, until a friend of mine called and asked if I could build him such a device for his slot-car racing group. He filled me in on the requirements, but cost was one question I couldn't answer until I searched the Web first to familiarize myself with what was available and if it would be less expensive to purchase a device or build one from scratch. I decided to build one myself.

Although very simple in design, the circuit in **Figure 1** does provide high speed fast input monitoring. To simplify construction and initial expense, I designed two-input boards. However, identical two-input circuit boards can be *multiplexed* up to five boards, offering a maximum of 10 inputs [where '0' = '10']. For monitoring three inputs, use only one input circuit of a second circuit board.

Also indicated on the **schematic** are optional input devices. Regardless of which input device is used, any delay on an activated input device will not interfere with any following inputs because the associated flip-flop to the input device is immediately set [disabled], while at the same time providing a high speed one-shot pulse to its counter input.

If two or more inputs are entered within the exact same time frame [~100 µS], then each of them will display the same number indicating a tie. For instance, in a three car race, if cars in slots 1 and 3 both cross the finish line within the same time frame (as described above), they will both show number 1 on their displays as a tie, and slot car 2 will display a number 2 as the final input.

The Pin Bone's Connected to the Flip-flop ...

Both input device segments, DV1-DV2, are identically configured and operate exactly the same electrically. Examining only one side with input segment DV1 and everything at a reset condition, the DV1 input goes to pin 1 of IC2a through diode D1. Flip-flop IC1a has a low on its Q pin 1 output, a high on its Q-bar pin 2 output, and DS1 is off. With IC3's count enable pin 2 low, it is enabled to receive, count, and store input pulses. Although the pin 3 display enable input is always enabled, the output number data [0] to the DS1 number display does not light because the ground providing FET [T1] to the display is not activated yet.

When DV1 is activated, a high input to pin 1 of IC2a appears and output pin 3 will immediately initiate a very fast low to high output pulse, causing IC2b output pin 4 to also go high and set the IC1a flip-flop.

During this latter action, a low to high to low pulse was very quickly passed through blocking diode D3 into the clock input pin 1 of counter IC3, which is typical of a one-shot action because IC1a — being immediately set — caused pin 2 of the Q-bar output to reverse polarity and remain low, while pin 1 Q output (also reversed) went to and remained high. Prior to this reversing action, IC3 had a single pulse input that was stored as a one count and — at the same time — enabled the DS1 display output by turning FET T1 on to allow DS1 the necessary ground.

The purpose for having FET T1 in the circuit is actually three-fold: first, to preserve battery energy; and second, to provide an off-state storage condition without adding a power switch. The third use is to provide a

PARTS LIST

ITEM	PART #/DESCRIPTION
All Jameco parts unless otherwise marked [**www.jameco.com**]	
ICs:	
IC1 = 4013 flip-flop	#12677
IC2 = 4011 quad two-input NAND	#12634
IC3 & 4 = 4026 decade counters	#353560
DISPLAYS: Two per board [**www.futurlec.com**]	
Seven-segment 1" CC digital display	#7SR10012AS [RED]
[All CC with pins 3/8 grounds]	#7SG10012AG [GREEN]
	#7SG10011AY [YELLOW]
	#7SB10012AB [BLUE]
RESISTORS:	
R1, R2, R3, R4 = 47K	#691260
RP1-RP2 = 330 ohms	#108572
FETS:	
T1, T2, T3 = BS170	#256031
DIODES:	
D1, D2, D3, D4 = 1N914A	#655269
POWER ACCESSORIES:	
Nine volt battery clip	#105794
Nine volt battery snap	#1949488
Reset switch	#174414
DV INPUT/OUTPUT optional selections	
Input switches	#187717
Phototransistor	#2129385
Reed switch	#171872
Buzzer	#196357
Male headers [.100 20 break-away]	#103369

Misc: Solder and equipment; nine volt battery; wire/wire wrap equipment; enclosure; test equipment; RCA plugs/RCA jacks.

Circuit boards available from author: Two-input/two-display board $12 each. Drilled, two-sided plated through, non-masked, component ready; S&H included.

ground ONLY to the active-input display. Even though a counter is outputting the data for the zero in the reset condition, the lack of a ground through FET T1 to the display did not show it [preserving battery energy].

With the next input pulse count (number 1), the FET is turned on to provide a ground to display the intended number 1. Note here that the number 0 is not stored within the counter on a reset if it's disabling the DE input at that time with a low input.

If the display enable is returned to a high, it would then output and display the number 0 on the next input clock pulse and not the intended number 1. Using FETs allows for the DE inputs to be constantly enabled and prevents this unwanted situation.

And a One, and a Two ...

Notice that regardless of which input device is activated, *all* counters receive and store the exact same input pulses, and they are counted and stored within each of them as the same value. For example, if DV1 receives the first input, a number 1 is shown on its associated display, but a count of 1 is also stored within all of the other counters. However, only the DV# active input will display its immediate counter-input order number.

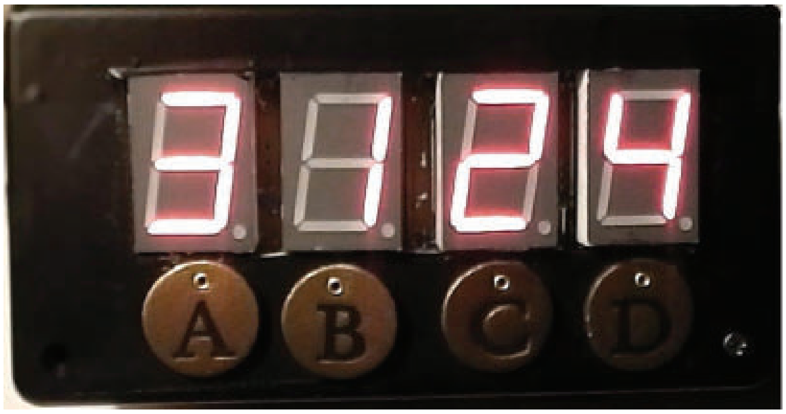
The very next device input will display a number 2 on its associated counter, even though all of the other counters have a 2 stored within them (but not displayed). Once a device input is made to its respective counter, the input order number is displayed but the chip enable pin 2 input is disabled, and prohibits any further inputs while continually displaying its initial input number.

Diodes D3 and D4 prevent other high output pulses to the clock inputs from being present on the AND gate output pins. As many as five boards can be ganged together by using jumper wires between each two-display boards, or by using two of the available four-display boards for eight display outputs.

What Else Ya Got?

Another use for this project might be to monitor (at the sound of a bell or light activation) the reaction input speed of pitching a ball at switch-activated targets among competing pitchers, or perhaps as an associated-type table game item.

To store this project, simply press the reset button to shut off all the displays and put it away. No negotiable power is consumed during this time. I recommend using a nine volt battery.



■ A look at the monitor.

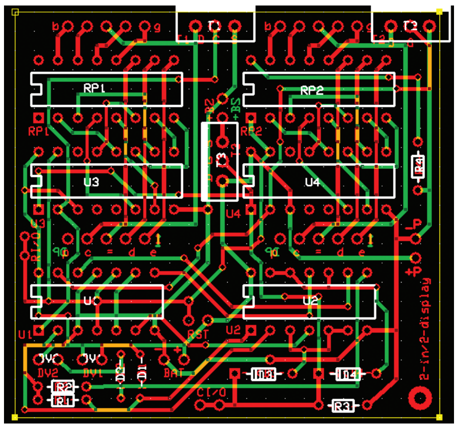
Putting It All Together

I constructed the project with four red displays [two boards jumpered together] in a 6" x 3" x 2" plastic enclosure. You can use any color display indicated in the Parts List; blue is the most expensive but yet the most visible. I recommend the use of a circuit board assembly.

I had no problems whatsoever constructing this project. I used jumpers between two two-input/two-display boards as indicated in the **schematic**. I combined two of these latter boards together for a single four-input/four-display board. If you use two two-display boards, I recommend that you solder in the .100 male headers [snapped off by two's] to easily place in each of the jumper locations, and use them to wire wrap between the board jumper connections.

As mentioned earlier, I do have four-input board layouts available, as well as the files for either board for those that want them. Again, simply email me and I will send the files for your use.

As a construction hint, solder all the resistors and



■ Two-input circuit board layout.

diodes in place on the board first, observing the polarities of the diodes. Next, solder the ICs, followed by the jumper terminals, then the FET transistors, and the battery snap wires with appropriate lengths to the off-board reset switch, input devices, and buzzer. Although I allowed for it, I didn't install a buzzer. If you prefer not to install one, then you don't have to solder FET T3 in place.

Now, insert the seven-segment displays onto the circuit board from the foil side. With this side of the board facing you, pin 1 [e] is indicated to the left and the seven-segment display should be inserted with the decimal point [DP] to the bottom right. Check all soldering connections, etc., before inserting the battery. The reset button is also the off button, and all displays will turn off for storage.

Making the Right Connections

If all wiring is correct, press the reset button. All displays should be off. Activate any of whatever device input medium is used, and a number 1 should appear on that input display. Repeated activation on any same input device should do nothing more.

Now, activate any other input device and a number 2 should appear on its associated display, and so on for all the other displays. Pressing the reset button clears all the displays; the unit is ready for more use or can be placed in storage.

The input device option that you choose will most likely determine your connection method. I would suggest using RCA plugs/jacks to provide for a solid and bounce-free connection. Not only would it be an easy connect/disconnect approach, but both are readily available with pre-assembled male-end cables in various lengths.

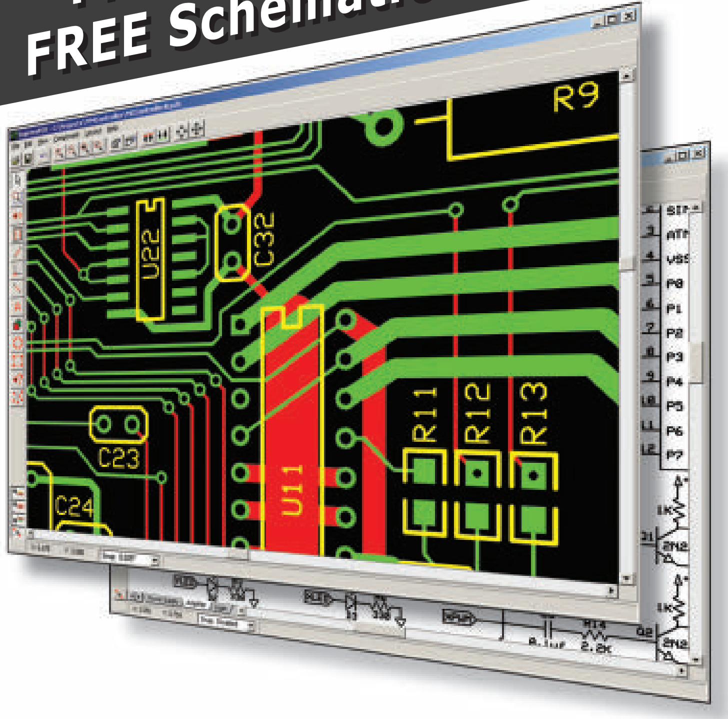
The Finish Line

Hopefully, you will have a fun application for this project and find many ways to use it.

Perhaps in a future article, I'll cover adding a timer feature if there is enough interest. Files to make your own circuit boards are available at the article link. Feel free to contact me with questions. **NV**

$51 For 3 PCBs

FREE Layout Software!
FREE Schematic Software!



01 DOWNLOAD our free CAD software
02 DESIGN your two or four layer PC board
03 SEND us your design with just a click
04 RECEIVE top quality boards in just days

expresspcb.com

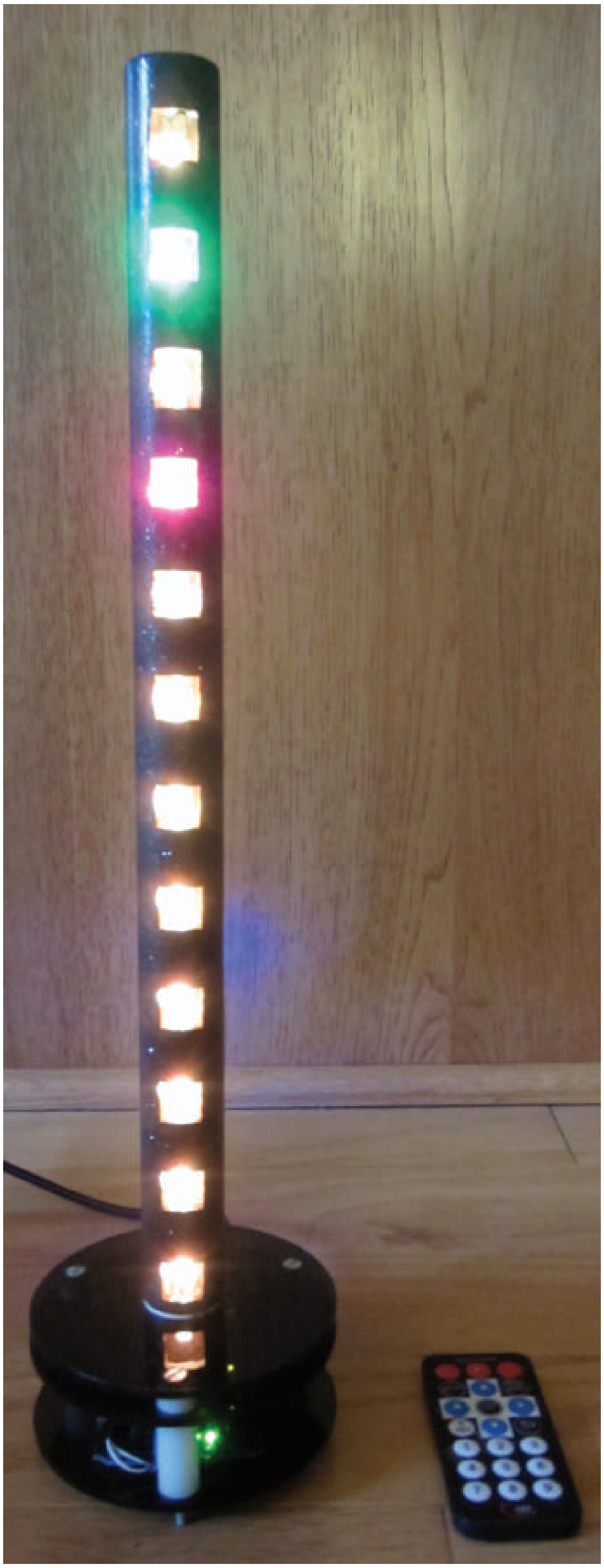
A UNIQUE LED CLOCK

By Craig A. Lindley
calhjh@gmail.com

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/march2014_Lindley.**

As hard as it might be to believe, I have never built an electronic clock of any kind. I've always thought electronic clocks were passe and not worth the time to design and build one.

In addition, I thought that all the really interesting electronic clock designs had already been built, so why bother. However, as I was looking around for something to do with the 15 feet of RGB LED ribbon I purchased from **Adafruit.com**, it occurred to me that I could use a short segment of the ribbon (14 LEDs' worth) to build a unique electronic clock which used the RGB LEDs to display the time and date, function as a mood light, and also run some animated bright and colorful patterns. I also decided to control the clock with an IR remote control so no physical access to the clock would be needed. Maybe designing and building an electronic clock could still be cool after all.



LED clock in operation.
As you can see, the LEDs are very bright! The time shown is between 9:55 and 9:59.

I already had an Arduino Uno microcontroller, an IR detector/sensor, and an IR remote control available, so the only part I was lacking was a real time clock (RTC). I decided to use a battery backed-up RTC to make the clock accurate, reliable, and impervious to power failures. I chose the ChronoDot ultra precise RTC module spec'ed at less than a minute of drift per year.

Designing the LED clock circuitry was easy because of the small number of parts involved. To simplify the design, I decided to power the clock via USB so no power supply components (internal to the clock) would be required. With all of the parts in hand, I breadboarded the circuitry and wrote the software for this unique LED clock using the Arduino 1.0.5 IDE (Integrated Development Environment) on my MacBook Pro.

The Clock Hardware

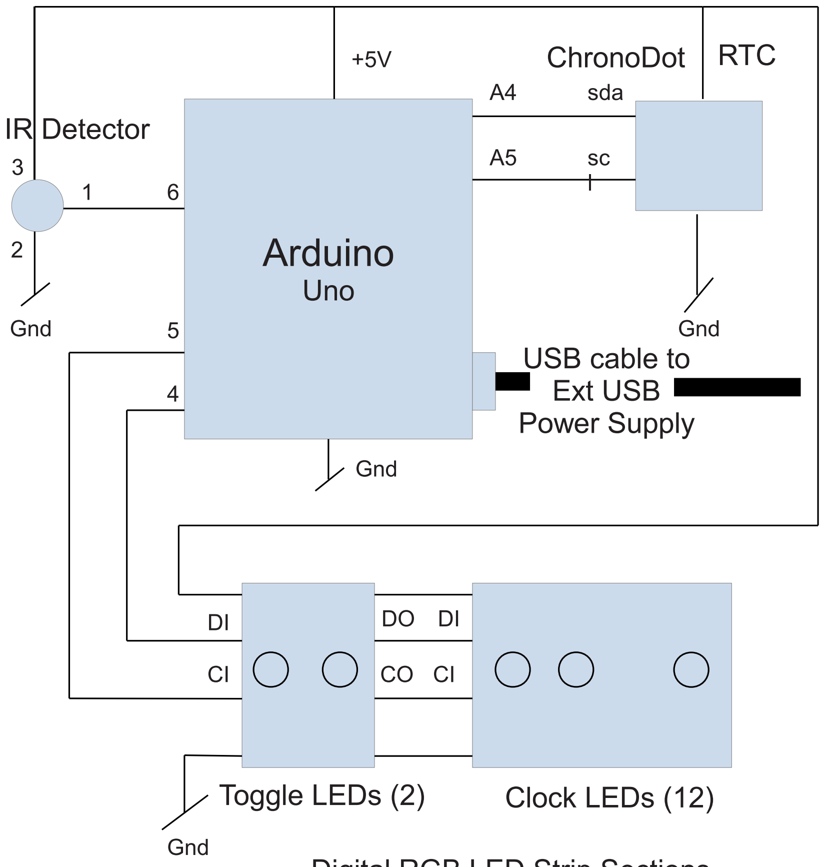
As mentioned, the hardware for this unique LED clock is quite simple and is shown in **Figure 1**. The clock is powered by an external USB power supply connected via a USB cable plugged into the Arduino Uno's USB connector. All of the clock's components run on the five volts provided by the Arduino Uno board.

The IR detector/sensor's output is connected to pin 6 of the Arduino which is configured as a digital input in software. The two Arduino digital outputs on pins 4 and 5 drive the RGB LED strip via some bit-banging performed in the software.

Finally, the ChronoDot RTC assembly is connected to the Arduino's I²C/TWI interface via pins A4 (SDA) and A5 (SCL). NOTE: If you use a different Arduino model for this clock, the pins for the I²C/TWI interface will almost certainly be different and must be taken into consideration.

All of the components are wired directly to the Uno board via inline connectors; no shield is involved, though you could use a prototyping shield if you like. Once you have the hardware wired up, insert the RTC battery into the socket. According to the specification, this battery should keep the RTC running for at least seven years.

That's about it for the hardware. As you may now realize, most of the clock's functionality is provided by the software.



Digital RGB LED Strip Sections
Adafruit ID: 306
Based on LPD8806 controller chips

FIGURE 1. Schematic diagram.

The Clock Software

The software consists of two files: *datatypes.h* and *AUniqueLEDClock.ino*. Both are available at the article link. These files must be moved into a directory called *AUniqueLEDClock* you create in the Arduino development area on your computer.

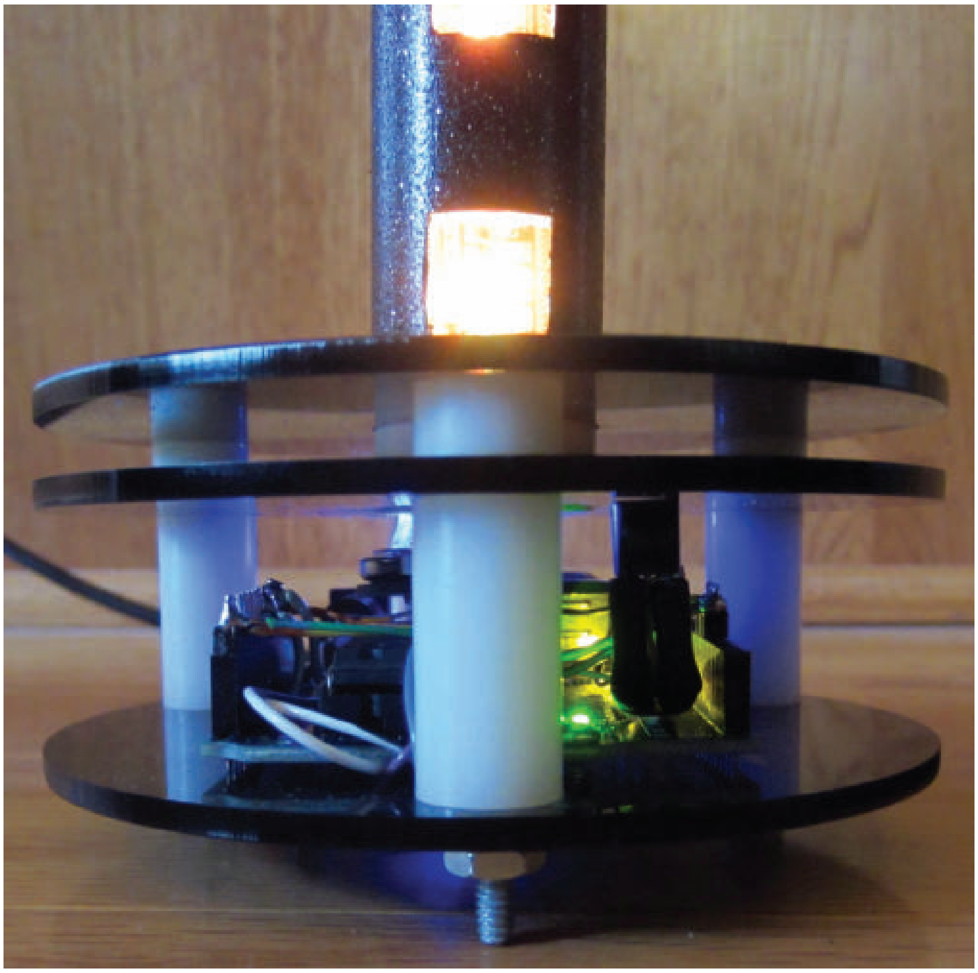
Two Arduino libraries are needed by the clock. The wire library for I²C/TWI communication and the IRremote library for remote control operation. The wire library is part of the standard Arduino IDE and doesn't require installation. The IRremote library, however, must be installed before being used. See **Resources** for info on how to get the library and install it. Code to control the ChronoDot RTC and the RGB LED strips is embedded in the *ino* (or sketch) file.

Once you have the clock files and have installed the IRremote library, you should be able to bring up the Arduino IDE and load the code. Make sure you select the Arduino Uno board type from the Tools menu or you will have issues. If you click the Verify button and don't see any errors, you should be able to click the Upload button to program your Arduino.

If all is well, the clock should go into the clock display mode (more on this in a bit) and you should be good to go.

The best way to understand the code is to examine the sketch file in detail. There isn't enough space in this article to give a detailed description of how the code works, but I will give you a quick rundown.

All Arduino sketches have two entry points: *setup*() and *loop*(). In our *setup*() function, all Arduino I/O pins are configured for interacting with the external hardware, the random number



■ **PHOTO 2. LED clock base detail.** Base made from three 4" smokey colored acrylic discs, some spacers, and long screws. IR detector is positioned vertically on the right of the white spacer, facing forward.

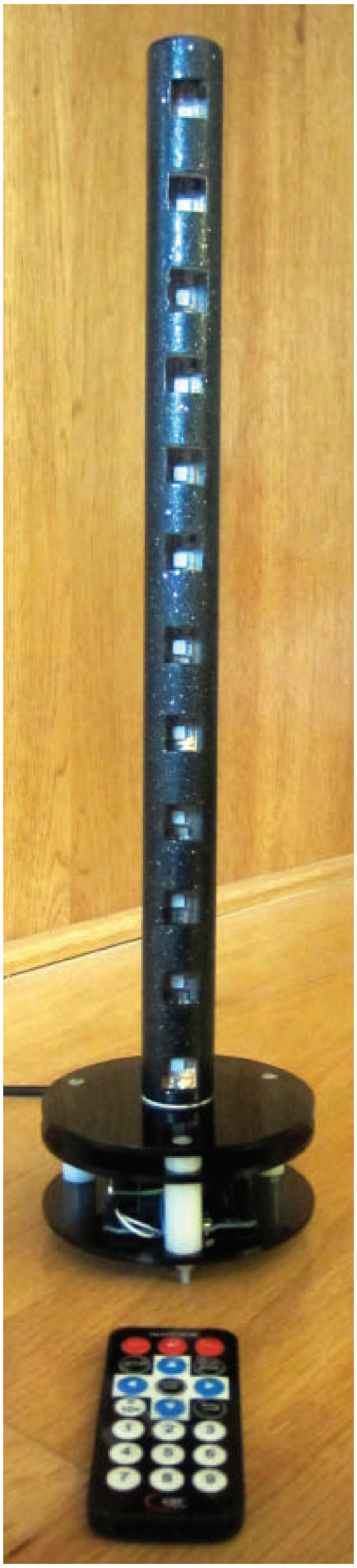
PARTS LIST

ITEM	SUPPLIER	COMMENTS
Arduino Uno microcontroller	eBay, **adafruit.com**, **mcmelectronics.com**, RadioShack, and many other places	Other Arduinos could be used but the sketch would need to change
ChronoDot ultra precise RTC ID: 255	**adafruit.com**	Comes with battery
Digital RGB LED strip portion* ID: 306	**adafruit.com**	Two LEDs in length for the toggling LEDs
Digital RGB LED strip portion* ID: 306	**adafruit.com**	12 LEDs in length for the clock LEDs
IR detector/sensor	RadioShack IR receiver #2760640 or Adafruit model TSOP38238 ID: 157	
21-key IR remote control ID: 389		Other remotes could be used but Arduino sketch would need to change
USB power supply		Must be capable of around one amp at five volts
USB to Arduino cable		Sometimes provided with Arduino Uno purchase

NOTES:

*LED strip is sold in one meter lengths of 32 LEDs. Since a total of 14 LEDs are used in this clock project, you can make two clocks with a one meter length and still have some left over for other projects.

Adafruit has a tutorial on how to safely cut the LED strip into sections. See **http://learn.adafruit.com/digital-led-strip/advanced-separating-strips**.



■ **PHOTO 3.** Here, you can see the RGB LEDs through square holes cut into the 1" PVC pipe. The PVC pipe was press-fit into a 1" hole drilled through the top two acrylic discs and painted with a dark gray sparkle paint.

generator is seeded, the wire library is initialized, and the IRremote library is enabled for reception of IR signals. A check is then made to be sure the RTC is running, then the Arduino's EEPROM is initialized for storage of the clock's color configuration information. Finally, all of the RGB LEDs are turned off until needed.

The *loop*() function is a lot more complex. In essence, it runs a series of state machines that control the operation of the clock. I chose a state machine implementation because it makes the code easier to read and maintain than if it were coded with massively nested *if-then-else* structures.

In a state machine, transitions between states occur when specific stimulus is applied while in a current state that is expecting it. All extraneous inputs are ignored. The input stimulus (in our case) results from keys being pressed on the remote control.

A good example of state machine operation can be seen in the function *hsvColorSetter* in the sketch. Here, clicking keys on the remote control causes the state machine to move back and forth between two states while the user configures a color for use in the clock. Clicking the STOP/MODE or the ENTER/SAVE keys on the remote causes the state machine to exit with the definition of the HSV color the user was setting.

HSV (or Hue, Saturation, and Value) color values are used extensively in the clock's code. Each of these color attributes can be manipulated independently and have different effects. Hue is the actual color of the color and is specified in degrees between 0 and 359. A hue of zero is red, a hue of 120 degrees is green, and a hue of 240 is blue. Hues in between these values are a mix of the surrounding colors.

Saturation determines how pure a color is and has values between 0.0 and 1.0. Fully saturated colors are deep and full, while lesser saturated colors trend towards pastels and white.

Value determines the brightness of the color and also ranges in value from 0.0 to 1.0. The higher the value, the brighter the color. As an example, a hue value of 0.0, a saturation value of 0.0, and a value of 1.0 results in the color white.

Inside the *loop*() function, the current time and date is read every second from the RTC and displayed on the LEDs if the clock is in clock mode. Other things happen in the other operational modes of the clock. See the discussion of the Remote Control for more info.

Reading the Clock

Reading the Time

I packaged my clock in a rather unique vertical format, so it takes a little time to get comfortable reading it. While displaying the time, all of the LEDs are first set to the background color (default blue), then the hour value is displayed (default red) with the top LED indicating 12 o'clock and counting down towards 1 o'clock at the bottom.

Since minutes are displayed on the same scale, they are only available with five minute resolution. To determine the minutes count, count up by fives from the bottom LED. For example, if the fifth LED from the bottom is lit (minutes are displayed in green by default), the minute count is between 25 and 30.

How often do you need the exact minute count? Not often, so a five minute resolution was deemed sufficient. Since hours and minutes are displayed on the same group of LEDs, there is often (once per hour) a conflict on which color to use. In these cases, the LED in conflict will blink back and forth between the hour and minute color.

For example, if the time is 10:50, the 10th LED from the bottom will blink red then green until the minute value changes, at which time there will again be one red LED for the hour and one green LED for the minutes displayed.

While time is being displayed, the two LEDs I refer to as the toggle LEDs alternate between the toggle color and off. The idea behind these LEDs is to simulate ticking of the clock.

Clock Events

If the clock did nothing but display time, it would get boring pretty quickly. To prevent boredom, I defined three events which fire periodically to liven things up. There is a 10 minute, a 15 minute, and a 30 minute event. With each 10 minute event, the time display is suspended and a beautiful rotating color wheel or rainbow effect is displayed.

With each 15 minute event, the clock goes from completely dark to brilliance blue in slowly increasing steps. With each 30 minute event, the full date is displayed. You have to see these events to appreciate them. The clock returns to its normal time display as soon as the event display is completed.

Reading the Date

The date will be displayed in a month/day/year sequence with unique colors for each quantity. Month (with a value of 1..12) maps directly to the LED display with January (or month one) at the bottom and December (or month 12) at the top. The day of the month (values 1..31) will be indicated by a rising dot in the day color, starting from the bottom LED and wrapping around the display as many times as necessary.

For example, the 29th of the month will loop through all of the LEDs twice (for a 24 count) and then light the fifth LED from the bottom to indicate the 29th. When displaying the year, the bottom LED indicates 2010 with each subsequent year one LED higher.

Remote Control

All aspects of the clock's operation are controlled via a 21 key IR remote control; there are no other buttons or switches to manipulate. The software I wrote for the clock embodies how I thought the remote control should work. You can change things if you so desire since you'll have access to the source code in the Arduino sketch which can be found at the article link.

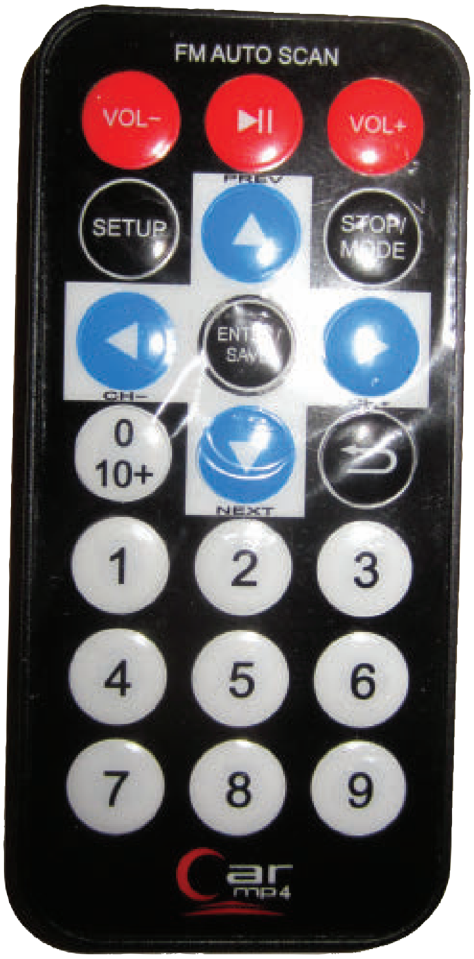


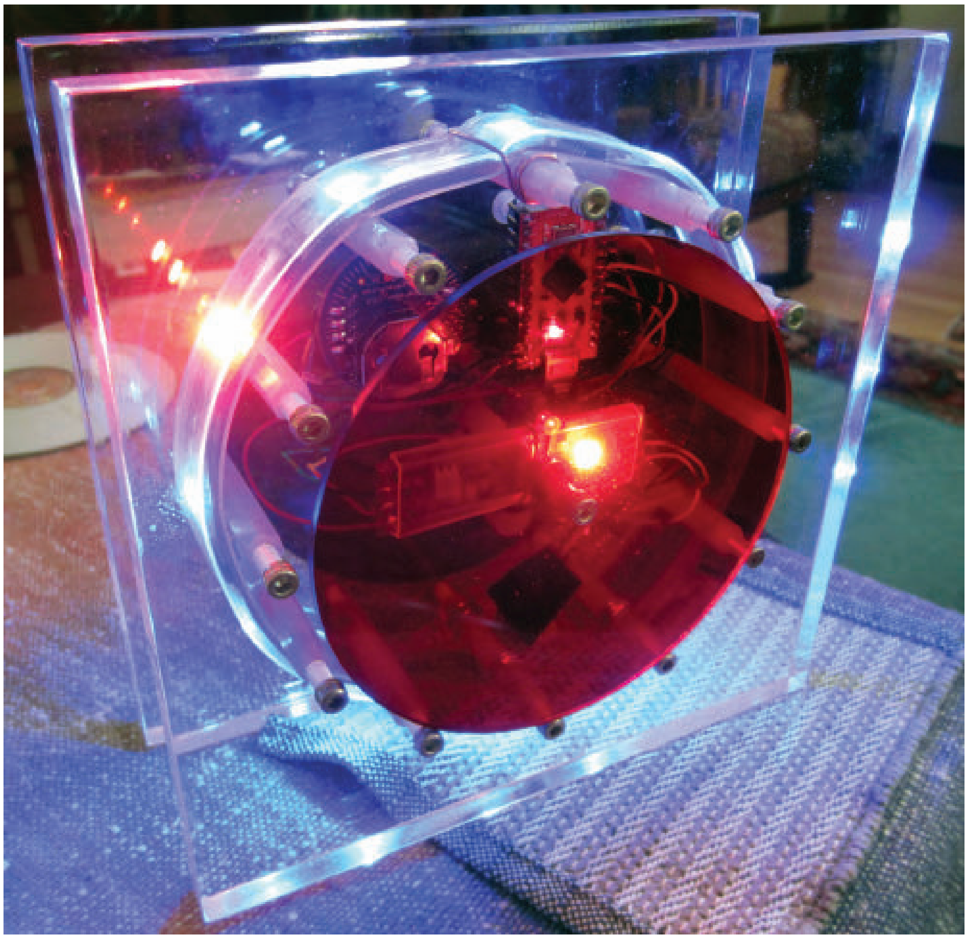
PHOTO 4. The IR remote control.

When the clock is first powered up, it immediately goes into clock mode even though the time or date has not yet been set. All colors used for display will be set to defaults coded into the software. NOTE: The clock will retain the colors you configure for each mode (even if the clock loses power) and will use these colors until changed again.

The clock will remain in clock mode until interrupted. Hitting the STOP/MODE key (the interrupt key) on the remote causes the clock mode to be exited and all LEDs to flash white indicating an interrupt has occurred and that you have the clock's attention. The clock now awaits your command input via the remote control. Here's a description of each key:

1 Key — Clicking this key causes the clock to go into clock mode. The clock will stay in clock mode displaying the time and date until interrupted.

2 Key — Clicking this key causes the clock to go into mood light mode where all of the LEDs display a mood



■ **PHOTO 5.** Alternative packaging using a SparkFun Pro micro Arduino.

light color of your choosing. Mood light color is altered using the 1 and 7 keys to change the color's hue up and down, respectively; the color's saturation is adjusted using the 2 and 8 keys; and the color's value or brightness is changed using the 3 and 9 keys.

The clock will remain in mood light mode until either the ENTER/SAVE or the STOP/MODE key is clicked. At that time, the clock will go back into interrupt mode awaiting a further command.

3 Key — Clicking this key puts the clock into pattern display mode. Without further user input via the remote, the clock will randomly choose one of its built-in display patterns to show and the speed at which to display the pattern. If you wish to change the speed of the pattern's animation, use the up and down arrow keys on the remote (up faster, down slower), followed by the ENTER/SAVE key to engage the new speed.

You can also select a different pattern for display by using the left and right arrow keys followed by the ENTER/SAVE key to select the new pattern. Clicking the STOP/MODE key interrupts the pattern display mode and moves the clock back into command mode.

4 Key — Click this key to set the clock's time. You first set the hour, then the 10s of minutes value, and then the minute value. Valid hour values are 1..12. Use the up and down arrow keys on the remote to set the hour value.

When the hour is correct, click the ENTER/SAVE key to proceed to setting the 10s of minutes value within the range of 0..5. A zero 10s of minutes value is indicated by no LEDs being lit. You must click the up key to set a non-zero 10s of minutes count.

When satisfied, click the ENTER/SAVE key to set the minute value within the range of 0..9. Clicking ENTER/SAVE after you have set the minute value puts the clock back into clock mode, which should display the time you have just set.

5 Key — Click this key to set the date. Setting the date is similar to setting the time except the values to set in order are month (1..12), 10s of days (0..3), day (0..9), 10s of years (1..9,) and year (0..9). After you set the date a couple times, you'll get the hang of it.

6 Key — Click this key to set the background color of the clock which is blue by default. Color is set as described for mood light mode.

7 Key — Click this key to set the colors associated with the time display. By default, the hour is displayed in red and minutes in green. You first set the hour color, then the minute color.

8 Key — Click this key to set the colors associated with the date display. By default, month is displayed in yellow, day in magenta, and year in cyan. You first set the month color, then the day color, then the year color.

9 Key — Click this key to set the color of the toggle LEDs. By default, the toggle LEDs display in red but you can change that to whatever color you desire.

Resources

The Arduino IRremote library was written by Ken Shirriff and is available at **https://github.com/shirriff/Arduino-IRremote**.

Details about the library can be found on Ken's blog at **www.righto.com/2009/08/multi-protocol-infrared-remote-library.html**.

Information about the ChronoDot can be found at **http://docs.macetech.com/doku.php/chronodot_v2.0**.

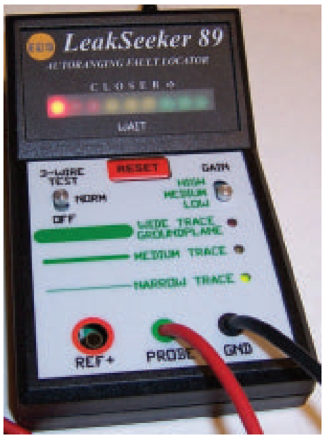
Information on the RGB LED strip can be found at **http://learn.adafruit.com/digital-led-strip**.

SETUP Key — When you click this key, all of your configured colors — for all modes — are erased and the default colors are reinstated.

Packaging the LED Clock

When I decided to build this clock, I wanted to make it unique looking as well as functional. In my first build, I made the clock out of acrylic discs and a vertical piece of PVC as shown in the **photos**. This resulted in a space-age/modern look which always takes people by surprise when I tell them it's a clock. When I decided I wanted one of these clocks for my desk, I decided to package it more like a "normal" clock. This version can be seen in **Photo 5**. You, of course, can package your clock anyway you like. **NV**

Locate shorted or leaky components or thermally defective components to the exact spot on any pcb, even multi-layer!



Your DVM shows the same shorted reading all along a shorted pcb trace because it doesn't have the resolution or range. The new ***LeakSeeker 89*** can find any short from zero to 300 ohms on any pcb, even multi-layer boards. Three fully automatic ranges!

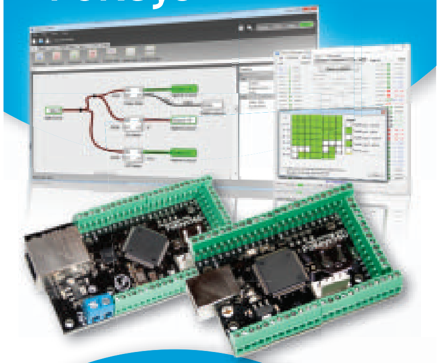
Watch the video: **Read the manual:**

Electronic Design Specialists, Inc www.eds-inc.com **561-487-6103**

Imagine this...

- a small and inexpensive
- USB or ethernet device
- with a rich set of features
- including PLC functionality and 3-axis stepper motor controller
- all accessible via free .NET, ActiveX or C++ library
- cross-platform
- configurable with free software

PoKeys



Or this...

- Smallest USB 2.0 portable 1MS/s oscilloscope
- Data acquisition of analog and digital signals
- Data recording
- Export to CSV, XLS, PDF and HTML
- Simple usage of advaced features
- Examples for C++, VB, Delphi and LabView
- Free software and updates

PoScope Mega1+

PoLabs

Visit us at
www.poscope.com

BREAKING THE ARDUINO SPEED LIMIT

By Bob Davis

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com /index.php?/magazine/article/march2014_Davis.**

Admit it, we all like to do it — break the speed limit. Usually, it is just a little bit. Sometimes it just keeps getting worse until we get caught. The same thing applies to overclocking our computers. We keep pushing the limits until it fries. How fast can you push the tiny little Arduino? Well, let's take a look.

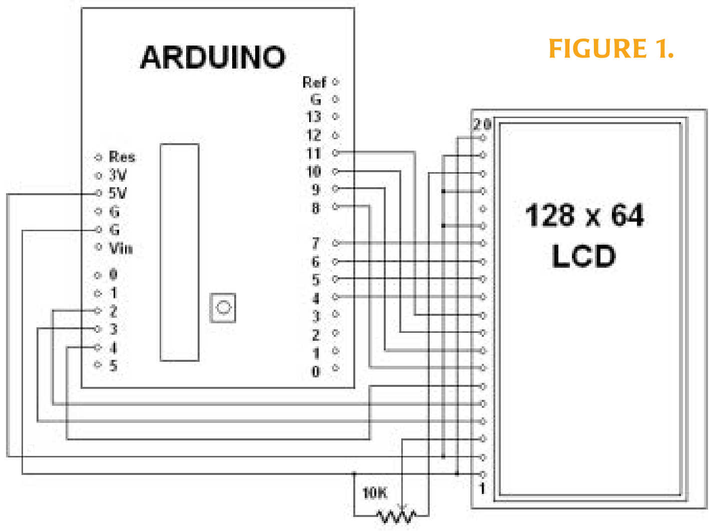


FIGURE 1.

First, we need to establish a base line. What are the default limits of the Arduino's analog-to-digital converter (ADC)? To determine the base line limit, we need to record the time; take, let's say, 100 samples; then record the time once again. Then, we subtract the beginning time from the ending time to see how long it took to collect those samples.

In order to easily see what we are doing, let's connect up an LCD screen. A 128x64 bit graphics screen is fairly cheap to buy, and is still able to display our data. The 128x64 LCDs typically sell for about $10 to $12 on eBay. Look for one with an ST7920 driver IC. You can use other driver ICs by changing a line of code in the software.

The 128x64 LCDs come in several sizes, and usually have a blue or green backlight. The green backlight seems to offer more contrast and makes it look more like a technical instrument. The larger 128x64 LCDs are 93 mm x 70 mm in size. They have a screen viewing glass area that is 72 mm x 40 mm. The LCDs also come with a variety of connectors. The best type to use is one that you can solder a 20-pin header onto and then plug it into a breadboard.

The ST7920-based LCD needs eight data bits and three control lines to run it. It will also need five volts and ground. You will need a variable resistor to set the contrast

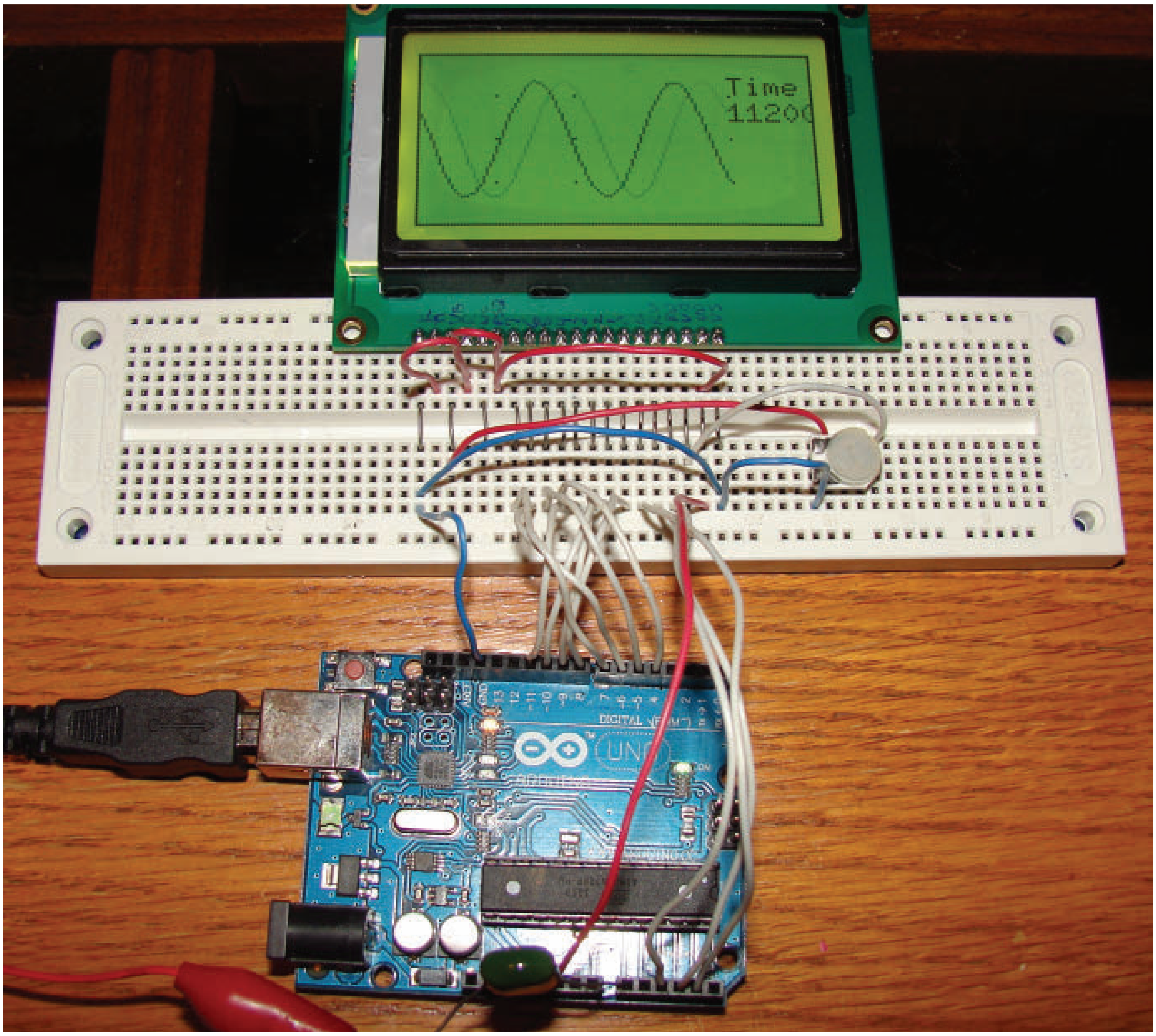


FIGURE 2.

of the LCD. **Figure 1** is the schematic diagram of how to connect the LCD screen to an Arduino Uno.

Figure 2 shows what that should look like. Note that I have jumpers over the center barrier of the breadboard. This arrangement makes some more space available for our jumper wires.

For a sine wave test source, I am using a program called "Audio SweepGen" running on my PC. To connect it to the Arduino analog input A0, there is a little bit of circuitry involved. **Figure 3** is the input circuit schematic. None of the resistor values are critical. You can sometimes get away with just using the .47 µF capacitor, as the Arduino's analog input seems to be naturally self-biasing to around 2.5 volts.

Here is the sketch to determine what the default conversion speed is:

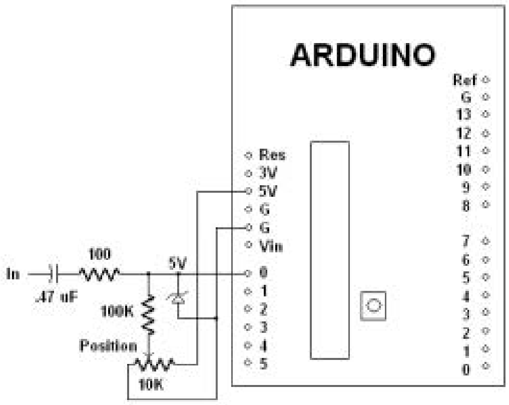


FIGURE 3.

```
/**********************************
128 by 64 LCD Oscilloscope - Default
By Bob Davis
Uses Universal 8bit Graphics Library,
http://code.google.com/p/u8glib/
  Copyright (c) 2012, olikraus@gmail.com  All
rights reserved.

*******************************************/
#include "U8glib.h"

// 8Bit Com: D0..D7: 8,9,10,11,4,5,6,7 en=18,
// di=17,rw=16
U8GLIB_ST7920_128X64_4X u8g(8, 9, 10, 11, 4, 5,
 6, 7, 18, 17, 16);

int Sample[128];
int Input=0;
int OldInput=0;
int StartSample=0;
int EndSample=0;
int SampleTime=0;

void u8g_prepare(void) {
  u8g.setFont(u8g_font_6x10);
  u8g.setFontRefHeightExtendedText();
  u8g.setDefaultForegroundColor();
  u8g.setFontPosTop();
}
void DrawMarkers(void) {
  u8g.drawFrame (0,0,128,64);
  u8g.drawPixel (25,16);
  u8g.drawPixel (50,16);
  u8g.drawPixel (100,16);
  u8g.drawPixel (25,32);
  u8g.drawPixel (50,32);
  u8g.drawPixel (100,32);
  u8g.drawPixel (25,48);
  u8g.drawPixel (50,48);
  u8g.drawPixel (100,48);
}

void sample_data(void){
// wait for a trigger of a positive going input
  while (Input < OldInput){
    OldInput=analogRead(A0);
    Input=analogRead(A0);
  }
// collect the analog data into an array
// do not do division by 10.24 here, it makes it
// slower!
  StartSample = micros();
  for(int xpos=0; xpos<100; xpos++) {
    Sample[xpos]=analogRead(A0);
  }
    EndSample = micros();
}

void draw(void) {
  char buf[12];
  u8g_prepare();
  DrawMarkers();
// display the collected analog data from array
// Sample/10.24 because 1024 becomes 100 = 5 volts
  for(int xpos=1; xpos<99; xpos++) {
    u8g.drawLine (xpos, Sample[xpos]/10.24, xpos+1,
    Sample[xpos+1]/10.24);
  }
  SampleTime=EndSample-StartSample;
  u8g.drawStr(100, 8, "Time");
  u8g.drawStr(100, 18, itoa(SampleTime, buf, 10));
}

void setup(void) {
  // assign default color value
  if ( u8g.getMode() == U8G_MODE_R3G3B2 )
    u8g.setColorIndex(255);     // RGB=white
  else if ( u8g.getMode() == U8G_MODE_GRAY2BIT )
    u8g.setColorIndex(3);        // max intensity
  else if ( u8g.getMode() == U8G_MODE_BW )
    u8g.setColorIndex(1);        // pixel on, black
}

void loop(void) {
// collect the data
  sample_data();
// show collected data
  u8g.firstPage();
  do { draw(); }
  while( u8g.nextPage() );
// rebuild the picture after some delay
  delay(100);
}
```

If you run this sketch, you will come up with a number around 11,200 microseconds as seen on the right side of the LCD display. That is a five digit number, so it barely fits on the screen.

We will be looking at two digit numbers before we are done. If you divide one by .0112 and then multiply that by 100 samples, you will get a speed of only around 8.9 thousand samples per second. **Figure 4** shows what the default screen looks like.

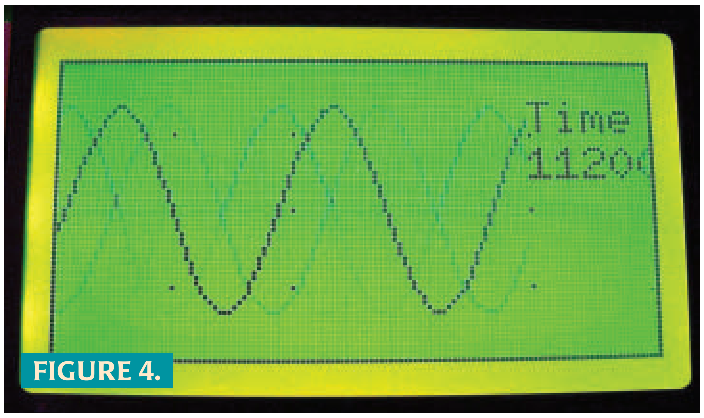


FIGURE 4.

Our next increase in speed will come from sacrificing some accuracy for that extra speed. The ADC is a 10-bit converter. If we don't need that much accuracy, we can speed up the converter. This can be done by changing the register that scales the clock down to a safe speed for the converter.

This register is called the *ADCSRA* register and the lowest three bits control the clock scaling. They are normally all set to ones.

We will clear the most significant of those three bits. To do that, right after *void loop(void)* { add these two lines of code:

```
// Clear bit 2 of ADC prescalar from 125KHz
// to 2 MHz
   ADCSRA &= ~(1 << ADPS2);
```

Now, upload the newly modified sketch and get ready for a shock. The time should have dropped to around 1,000 microseconds. We have sped up the converter by a factor of 16, but the other instructions kept us from seeing a complete 16 fold increase. If you divide 1 by .001 and multiply that by 100 samples, we are now at around 100

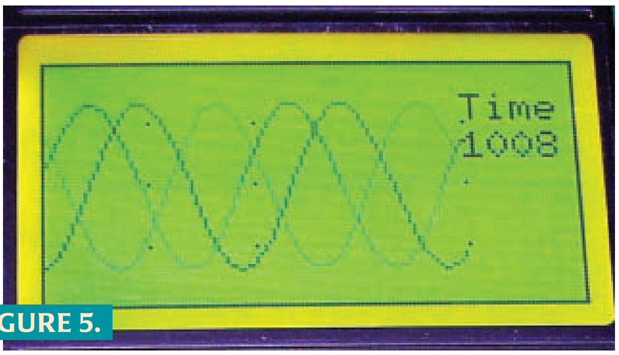


FIGURE 5.

thousand samples per second. **Figure 5** shows this screen with that modification.

Do you want some *more* speed? After reading the ADC, the Arduino software shuts it down. We can turn it back on by adding this next code, right after we take the sample with an *analogRead (A0):*

```
ADCSRA |= (1 << ADSC);    // Restart AtoD Conversion
```

That should further reduce the time to get 100 samples to 860 microseconds. That is 116,000 samples per second. There are some other tweaks available that might get a little more speed out of the ADC, but these two are the easiest to understand and make the most difference.

Can we keep on speeding it up? Yes, we can! However, we have reached the limits of the internal ADC. So, next we will look at "digital" inputs for a short while before adding an external ADC.

The tricky part will be getting all of the pins of one of the Arduino ports available. To do that, we need to move the three wires on the analog pins A2, A3, and A4 to the digital pins 3, 2, and 1.

Then, we need to change the driver configuration in our code to match. We will replace the line *U8GLIB_ST7920_128X64_4X u8g(8, 9, 10, 11, 4, 5, 6, 7, 18, 17, 16);* with *U8GLIB_ST7920_128X64_4X u8g(8, 9, 10, 11, 4, 5, 6, 7, 1, 2, 3);*. Then, upload and run the modified sketch to see if everything still works.

If everything went well, it should still be working. Now, we have all six of the analog pins available. We can use the pins as a logic analyzer with these two changes in our code.

First, the sample collection routine needs to be changed from analog to digital:

```
void sample_data(void){
// wait for a trigger of a positive going input
  while (digitalRead(A0)==0) {  }
// collect the analog data into an array
  StartSample = micros();
  for(int xpos=0; xpos<100; xpos++) {
    Sample[xpos]=digitalRead(A0);
  }
    EndSample = micros();
}
```

Next, the draw routine needs to be changed to digital, as well:

```
void draw(void) {
  char buf[12];
  u8g_prepare();
  DrawMarkers();
// display the collected analog data from array
  for(int xpos=1; xpos<99; xpos++) {
    u8g.drawLine (xpos, Sample[xpos]*16+8, xpos+1,
    Sample[xpos+1]*16+8);
  }
  SampleTime=EndSample-StartSample;
  u8g.drawStr(100, 8, "Time");
  u8g.drawStr(100, 18, itoa(SampleTime, buf, 10));
}
```

Using *digitalRead* instead of *analogRead* should have reduced your time to 424 microseconds. That is about twice as fast as the fastest analog read was. Using our math formula, the sample rate is now 232,000 samples per second. **Figure 6** shows a screenshot.

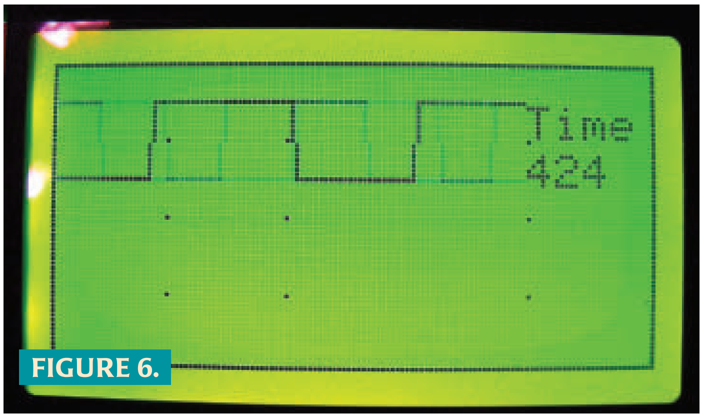


FIGURE 6.

Next, we can replace *digitalRead* with an even faster way of reading our data. This faster method is to use a parallel input port C or PINC command. That command looks like this: *Sample[xpos]=PINC;*. Change your code by replacing *digitalRead(A0);* with *PINC;* then upload it. You will now get a whopping data collection time of 80 microseconds! Now, we are looking at a two digit number. That is 1.25 million samples per second.

Are we going fast enough yet? Heck no! We can do MUCH better than that! First, do you want to use some of those other analog input pins? Since we are using the parallel input command, we have collected data from all of the analog pins. We just need to display that data.

This gets a little hairy as we have to use Boolean bit masking to pull out the individual bits. To do that, you will need to replace the one line *drawLine* routine with these six lines of code:

```
    u8g.drawLine (xpos, ((Sample[xpos]&
    B00000001)*4)+4, xpos,
((Sample[xpos+1]&B00000001)*4)+4);
    u8g.drawLine (xpos,
((Sample[xpos]&B00000010)*2)+14, xpos,
 ((Sample[xpos+1]&B00000010)*2)+14);
    u8g.drawLine (xpos,
((Sample[xpos]&B00000100)*1)+24, xpos,
 ((Sample[xpos+1]&B00000100)*1)+24);
    u8g.drawLine (xpos, ((Sample[xpos]&
    B00001000)/2)+34, xpos,
((Sample[xpos+1]&B00001000)/2)+34);
    u8g.drawLine (xpos,
((Sample[xpos]&B00010000)/4)+44, xpos,
 ((Sample[xpos+1]&B00010000)/4)+44);
    u8g.drawLine (xpos,
((Sample[xpos]&B00100000)/8)+54, xpos,
 ((Sample[xpos+1]&B00100000)/8)+54);
```

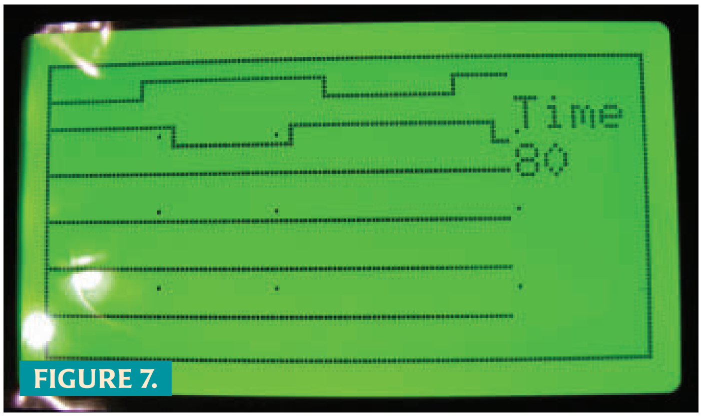


FIGURE 7.

Figure 7 shows the LCD screen in logic analyzer mode.

Now, back to speeding things up a bit. Observe that when using integers, we are inputting two bytes of data every time we read the C port. What happens when we replace *int Sample[128];* with *byte Sample[128];*? Try it and take a look.

We are now at 53 microseconds for 100 samples. That is two million samples per second. Can we go faster? Oh ya.

We call the next speed trick the "verbose" method. Using a loop limits our speed because of the time that it takes to loop back. That can be a long time. If we eliminate the "back tracking" and just go forward, we will be much faster.

Use the following code for your sample data collection:

```
void sample_data(void){
// wait for a trigger of a positive going input
  while (digitalRead(A0)==0) {   }
// collect the analog data into an array
  StartSample = micros();
  Sample[1]=PINC;  Sample[2]=PINC;  Sample[3]=PINC;
  Sample[4]=PINC;  Sample[5]=PINC;  Sample[6]=PINC;
  Sample[7]=PINC;  Sample[8]=PINC;  Sample[9]=PINC;
  Sample[10]=PINC; Sample[11]=PINC;  Sample[12]=PINC;
  Sample[13]=PINC; Sample[14]=PINC;  Sample[15]=PINC;
  Sample[16]=PINC; Sample[17]=PINC;  Sample[18]=PINC;
  Sample[19]=PINC; Sample[20]=PINC;  Sample[21]=PINC;
  Sample[22]=PINC; Sample[23]=PINC;  Sample[24]=PINC;
  Sample[25]=PINC; Sample[26]=PINC;  Sample[27]=PINC;
  Sample[28]=PINC; Sample[29]=PINC;  Sample[30]=PINC;
  Sample[31]=PINC; Sample[32]=PINC;  Sample[33]=PINC;
  Sample[34]=PINC; Sample[35]=PINC;  Sample[36]=PINC;
  Sample[37]=PINC; Sample[38]=PINC;  Sample[39]=PINC;
  Sample[40]=PINC; Sample[41]=PINC;  Sample[42]=PINC;
  Sample[43]=PINC; Sample[44]=PINC;  Sample[45]=PINC;
  Sample[46]=PINC; Sample[47]=PINC;  Sample[48]=PINC;
  Sample[49]=PINC; Sample[50]=PINC;  Sample[51]=PINC;
  Sample[52]=PINC; Sample[53]=PINC;  Sample[54]=PINC;
  Sample[55]=PINC; Sample[56]=PINC;  Sample[57]=PINC;
  Sample[58]=PINC; Sample[59]=PINC;  Sample[60]=PINC;
  Sample[61]=PINC; Sample[62]=PINC;  Sample[63]=PINC;
  Sample[64]=PINC; Sample[65]=PINC;  Sample[66]=PINC;
  Sample[67]=PINC; Sample[68]=PINC;  Sample[69]=PINC;
  Sample[70]=PINC; Sample[71]=PINC;  Sample[72]=PINC;
  Sample[73]=PINC; Sample[74]=PINC;  Sample[75]=PINC;
  Sample[76]=PINC; Sample[77]=PINC;  Sample[78]=PINC;
  Sample[79]=PINC; Sample[80]=PINC;  Sample[81]=PINC;
  Sample[82]=PINC; Sample[83]=PINC;  Sample[84]=PINC;
  Sample[85]=PINC; Sample[86]=PINC;  Sample[87]=PINC;
  Sample[88]=PINC; Sample[89]=PINC;  Sample[90]=PINC;
  Sample[91]=PINC; Sample[92]=PINC;  Sample[93]=PINC;
  Sample[94]=PINC; Sample[95]=PINC;  Sample[96]=PINC;
  Sample[97]=PINC; Sample[98]=PINC;  Sample[99]=PINC;
  EndSample = micros();
}
```

We have now reached a limit that I have not broken yet. Believe me, I've been trying. We are now at 20 microseconds to collect 100 samples, or FIVE million samples per second. We have gone from 11,000 to 20 microseconds sample time, which is over 500 times faster. That is about as fast as the Arduino gets.

Figure 8 is the screenshot, with a 20 KC signal being displayed.

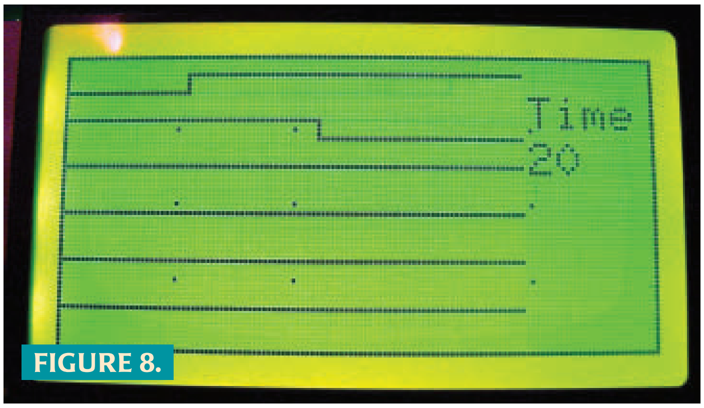


FIGURE 8.

Adding the External ADC

Next, we need to look into adding an external analog-to-digital converter.

There are many things to consider in buying an external ADC. Does it need to fit a breadboard without an adapter? What is the maximum conversion speed that we will need? How many support chips do we want to add? What is the input voltage range we want to monitor?

Many external ADCs are limited to one or two volts peak-to-peak of the analog input signal. Some require that the input be biased at up to four volts. Some converters require a negative power supply. How many bits do we need? Some converters can do six bits, some eight bits, some 10 bits, and some are even 12 bits or more.

I settled on a CA3306 six-bit 15 MC ADC. It is cheap, readily available on eBay, and it has 18 DIP pins that plug easily into a breadboard. For support, the CA3306 only needs three capacitors and a 5-15 MC clock source.

Figure 9 is the CA3306 schematic diagram, modified a little bit from the specifications manual.

Figure 10 is the clock schematic diagram; you can use any oscillator from 5 MC to 15 MC. I have even tested it out using up to 25 MC with no problems. However, if you try to connect it to the Arduino clock on pins 9 or 10, the Arduino will slow down or stop.

Once the CA3306 is wired up, you should see a logic analyzer displaying the outputs of it. You need to change the six lines of code for the logic analyzer back to an oscilloscope type display like this:

```
u8g.drawLine (xpos, Sample[xpos], xpos+1,
 Sample[xpos+1]);
```

If you are using a 20 KC sine wave, you will only be seeing 1/4 of the waveform. You can slow it down by replacing the 100 PINCs with the older data sampling version that used a loop and PINC.

Figure 11 shows the CA3306 ADC. The clock

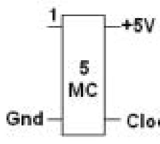


FIGURE 10.

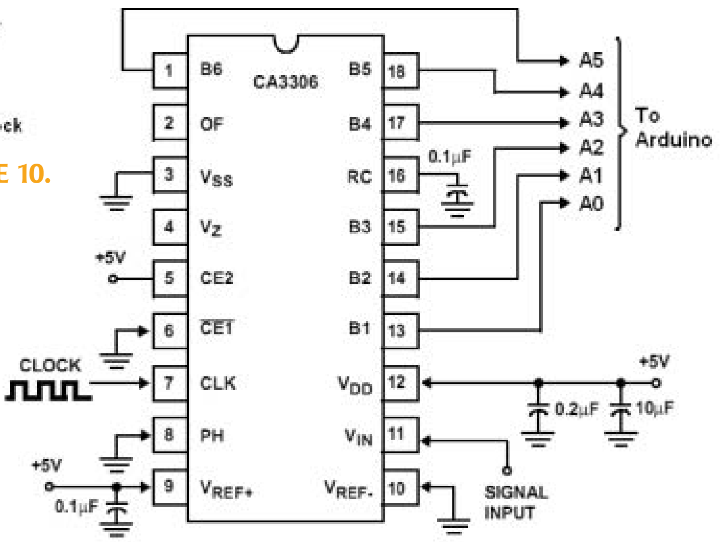
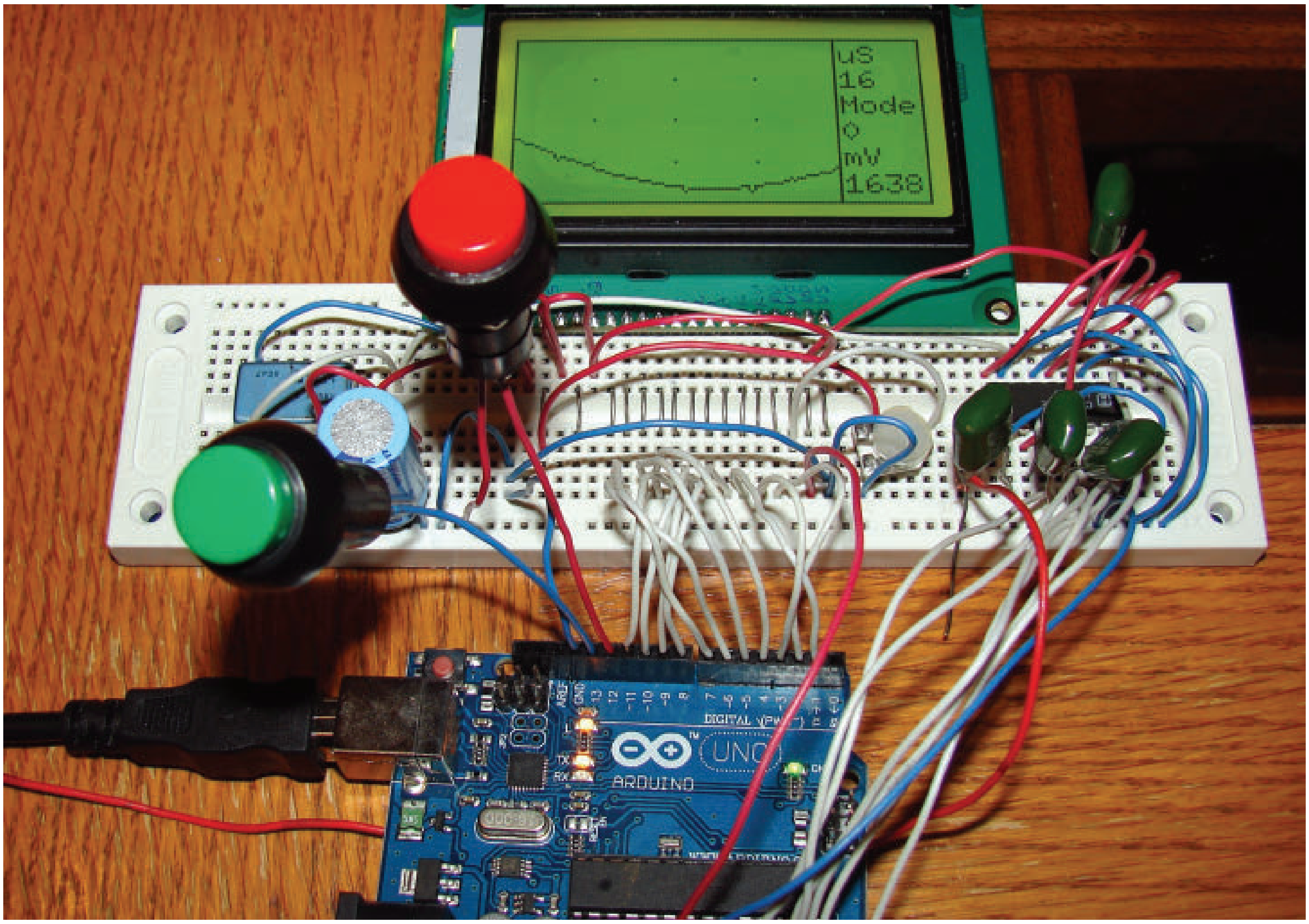


FIGURE 9.

FIGURE 11. The CA3306 analog-to-digital converter is on the right side and the crystal oscillator is on the left side behind the switches.



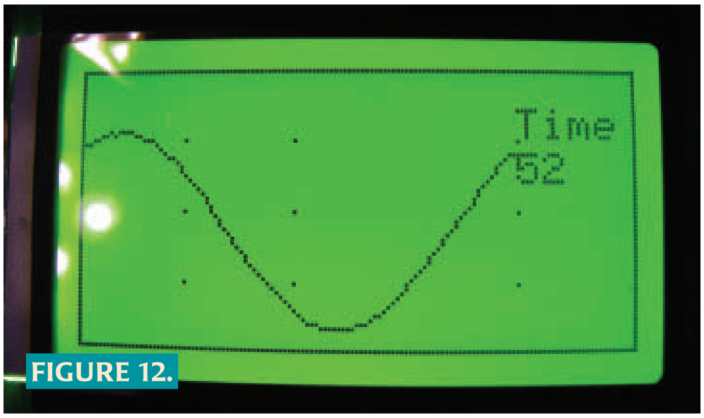


FIGURE 12.

oscillator was located at the other end of the breadboard. **Figure 12** is the LCD with the external ADC.

If you connect the analog input circuit that was shown earlier to the CA3306, the position control will not work very well. The reason is that the CA3306 has low input impedance. To compensate for that, we could add a better analog input section. An LF353 or TL082 dual op-amp can provide gain up to about 4 MC and over one million ohms of input impedance, as well as fixing the position control issue. The drawback is that it requires a positive and negative nine to 12 volt power supply. Two nine volt batteries will work. If you are building a portable wave form viewer, you may already have the Arduino running on a nine volt battery, so adding a second one might not be as big of a problem. **Figure 13** is a simple analog input schematic diagram.

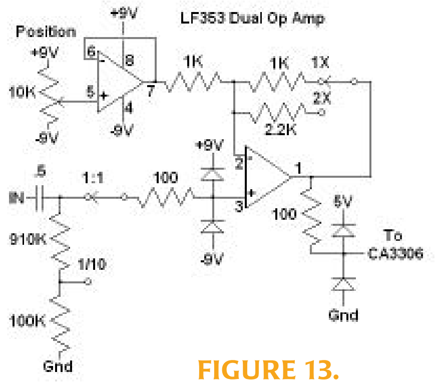


FIGURE 13.

Next, the switches are connected from D12 and D13 to ground. They are scan frequency "up" and "down" buttons. Then, the software can select between the verbose method and the loop method for collecting the data. Once using the loop method, a *delayMicroseconds (dtime);* command can be added. The delay value can then rotate between 1, 5, 10, 50, 100, 500, 1,000, and 5,000 at the push of a button. The complete code for the

WORLD'S MOST VERSATILE
CIRCUIT
BOARD HOLDERS

Model 324

Our Circuit Board Holders add versatility & precision to your DIY electronics project. Solder, assemble & organize with ease.

Model 201

VISIT US ON

MONTHLY CONTEST

Visit us on Facebook® to post a photo of your creative PanaVise project for a chance to win a PanaVise prize package.

PANAVISE®
Innovative Holding Solutions

7540 Colbert Drive • Reno • Nevada 89511 | (800) 759-7535 | www.PanaVise.com

AP CIRCUITS
PCB Fabrication Since 1984

As low as...
$9.95
each!

Two Boards
Two Layers
Two Masks
One Legend

Unmasked boards ship next day!

www.apcircuits.com

VISA MasterCard PayPal IPC MEMBER ASSOCIATION CONNECTING ELECTRONICS INDUSTRIES BBB final version is available at the article link.

Figure 14 is a screenshot of the completed oscilloscope. The top right shows the time it took to collect 100 samples; below that is the sample mode selection, and at the bottom is the millivolt reading based on an ADC that takes a five volt input for a full scale reading.

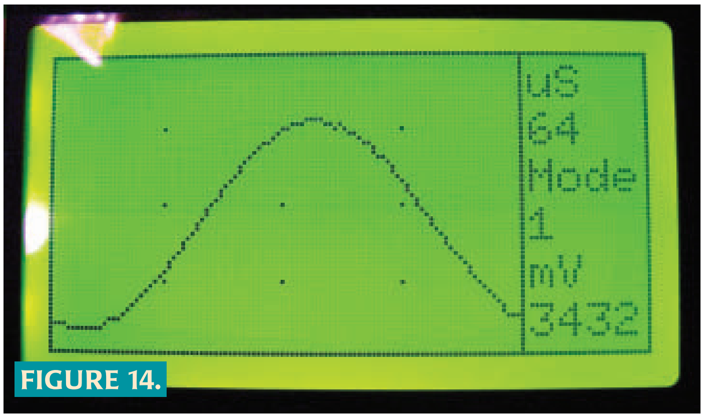


FIGURE 14.

That's a Wrap

Are faster speeds possible? Yes! With an AD775 25 MC ADC and a DS2010 FIFO, I have reached speeds of 25 million samples per second. A FIFO is a memory device that features a memory organization that results in "First In First Out." You can record data to a FIFO at its maximum clock speed, then play it back at much slower speeds. I suspect that with the right ADC and FIFO you could reach 100 million samples per second. Is that practical? No. If you add a FIFO, it then controls the sampling speed. So, you would also need to add two or three 74LS390s and a switch to select the sampling speed. At this point, the Arduino is controlling almost nothing other than the LCD.

If you were using an Arduino Mega, you would have enough pins to add a 74LS138 or two as frequency selectors, and to control relays for the input attenuation and gain switches. In fact, you would have enough pins to add a second ADC and FIFO to make a dual-channel 100 million samples per second oscilloscope. The Arduino Uno lacks the data pins that are needed to make a really good oscilloscope. **NV**

The Convenient All-in-One Solution
for Custom-Designed Front Panels & Enclosures

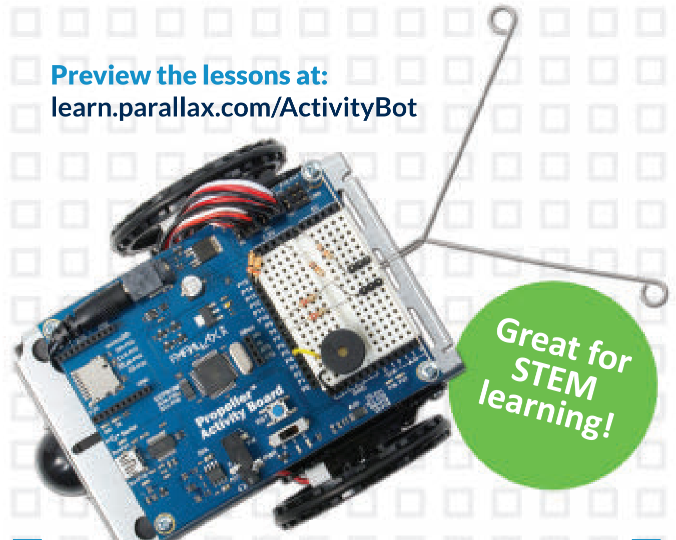
You design it
to your specifications using our FREE CAD software, Front Panel Designer

We machine it
and ship to you a professionally finished product, no minimum quantity required

- Cost effective prototypes and production runs with no setup charges
- Powder-coated and anodized finishes in various colors
- Select from aluminum, acrylic or provide your own material
- Standard lead time in 5 days or express manufacturing in 3 or 1 days

FRONT PANEL EXPRESS
FrontPanelExpress.com
1(800)FPE-9060

Preview the lessons at:
learn.parallax.com/ActivityBot



[ActivityBot]
ROBOT KIT $199

Learn to program in C on Mac or PC. Online tutorials are suited for the self-guided learner yet easily expand to accomodate a classroom or club setting.

PARALLAX INC
www.parallax.com

MakerPlot – Part 6
The DIY Software Kit

By John Gavlik
and Martin Hebel

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/march2014_MakerPlot**.

This is the concluding article on our MakerPlot bi-directional monitoring and control series. Part 5 dealt with using an Arduino Uno to control a setpoint level for a potentiometer value, where MakerPlot displayed the actions of two physical pushbutton switches that are attached to the Arduino in order to change the setpoint level up or down. While we showed you how the Arduino can also output MakerPlot commands as well as analog and digital data to do this, it was still really a one-way communications example as in micro-to-MakerPlot and not the other way around.

This time, we're going to show you how MakerPlot can go in the other direction to send back information to the Arduino in order to control the firmware's setpoint value using a **Slider** control on the Interface screen. All of this capability is embedded in the Arduino's code using MakerPlot instructions, so it's a very interesting way to look at how to interact with the micro instead of using a physical front panel with knobs and meters and such. (If you haven't already done so, you can download a free 30 day trial copy of MakerPlot from **www.makerplot.com** to follow along.) Let's get going.

The Basics of MakerPlot Bi-Directional Control

Figure 1 illustrates the intended bi-directional interactions between the Arduino and MakerPlot. In simple terms, we'll show you how to program the Arduino to ask for **Slider** information from MakerPlot, and then go on to show you how MakerPlot responds to this request via the same serial link it uses to send analog and digital data.

Once the **Slider** value from the MakerPlot Interface screen is acquired, the Arduino sketch can then "fine-tune" this value using the SW1 and SW2 pushbutton switches.

Finally, the fine-tuned **Slider** value is sent back to MakerPlot where you can see the value change as you push either of the SW1 or SW2 buttons. What

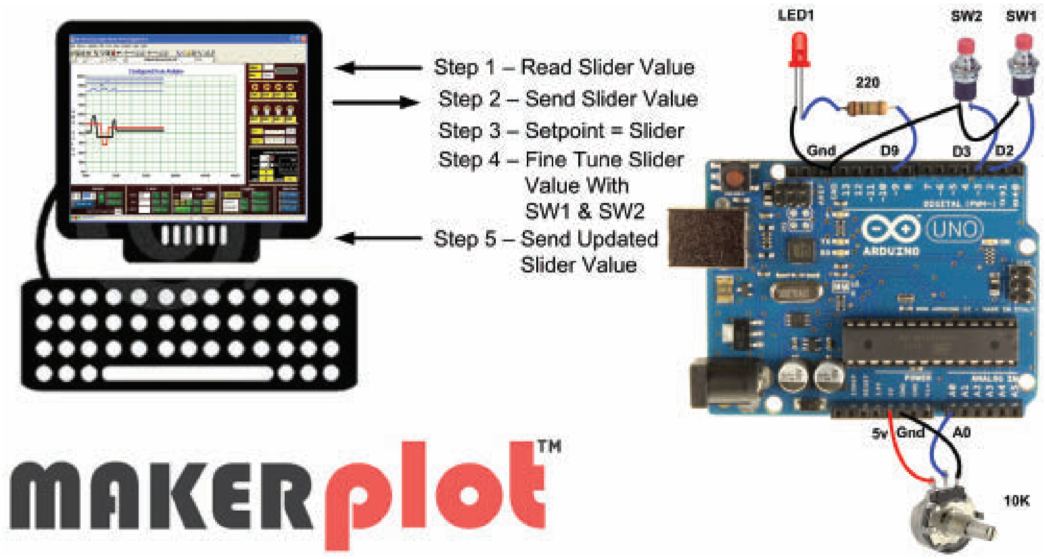

Step 1 – Read Slider Value
Step 2 – Send Slider Value
Step 3 – Setpoint = Slider
Step 4 – Fine Tune Slider Value With SW1 & SW2
Step 5 – Send Updated Slider Value
LED1
SW2
SW1
220
Gnd
D9
D3
D2
UNO
ARDUINO
5v
Gnd
A0
10K
MAKERplot™

Figure 1. MakerPlot bi-directional data exchange.

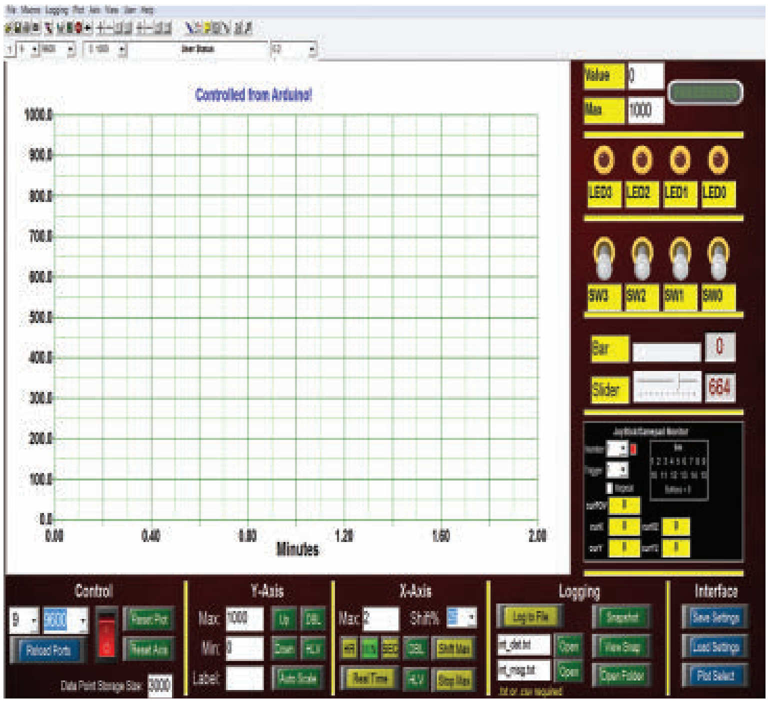


Figure 2. Interactive Interface for Setpoint Example.

this means is that you can start by adjusting the setpoint value with the MakerPlot **Slider** control with your mouse, then proceed to fine-tune this value with the SW1 and SW2 pushbutton switches. All of this is accomplished in the Arduino sketch using MakerPlot instructions; let's see how this is done.

Coding for Bi-Directional Data Acquisition and Control

We're going to continue to use the MakerPlot **Interactive Interface** (**Figure 2**) from Part 5 since it's the one that's specifically designed for this kind of experimentation and, also, it's the one Interface in the MakerPlot off-the-shelf suite that has a **Slider** control. **Figures 3** and **4** represent the Arduino

```
// MakerPlot Arduino Bi-Directional Control - Part 2
// This sketch is in the public domain

const int analogInPin = A0;   // Analog input pin that the potentiometer is attached to
const int LEDpin = 9;         // Analog output pin that the LED is attached to
const int SW1pin = 2;         // Pushbutton switch, raise setpoint
const int SW2pin = 3;         // Pushbutton switch, lower setpoint

int sensorValue = 0;        // value read from the pot
int setPoint = 100;        // Initial value of setPoint
int SW1state = 0;          // Store state of SW1
int SW2state = 0;          // Store state of SW2
int LEDstate = 0;          // Store state of LED

void setup() {
  // configure hardware
  pinMode(SW1pin, INPUT_PULLUP);  // Enable pull-ups on switches
  pinMode(SW2pin, INPUT_PULLUP);
  pinMode(LEDpin, OUTPUT);       // set LED to be an output pin
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);

  delay(2000);                              // allow connection to stabilize
  Serial.println();                   // send in case garbage in queue
  Serial.println("!RSET");                  // Reset the plot
  Serial.println("!O butMin=1");            // Set butMin object to 1, ON
  Serial.println("!O butMin.Run");          // Run event code button  to configure minutes, turn other 2 off
  Serial.println("!O txtXMax=2");           // Set text box for maximum time to 2 minutes
  Serial.println("!O txtXMax.Run");         // Run text box event code to update time
                                  // clear constant drawings and place constant text on plot
  Serial.println("!CLRC(CR)@TEXT 30A,105A,1.5A,(Blue),Controlled from Arduino!");

  delay(3000);                         // delay 3 seconds

  Serial.println("!O LED0=1");               // turn ON LED0 on the interface
  Serial.println("!O SW0=1");                // turn ON SW0 on the interface
  Serial.println("!BELL");                 // sound PC Bell
  delay(1000);                         // delay 1 second

  Serial.println("!O LED1=1");               // turn ON LED1 on the interface
  Serial.println("!O SW1=1");                // turn ON SW1 on the interface
  Serial.println("!BELL");                 // sound PC Bell
  delay(1000);                         // delay 1 second

  Serial.println("!O LED2=1");               // turn ON LED2 on the interface
  Serial.println("!O SW2=1");                // turn ON SW2 on the interface
  Serial.println("!BELL");                 // sound PC Bell
  delay(1000);                         // delay 1 second

  Serial.println("!O LED3=1");               // turn ON LED3 on the interface
  Serial.println("!O SW3=1");                // turn ON SW3 on the interface
  Serial.println("!BELL");                 // sound PC Bell
  delay(1000);                         // delay 1 second

  Serial.println("!O LED*=0");               // turn OFF all LEDs
  Serial.println("!O SW*=0");                // turn OFF all Switches
  Serial.println("!BELL");                 // sound PC Bell
  delay(1000);                         // delay 1 second
}
```

Figure 3. Part 1 of bi-directional setpoint sketch.

```
void loop() {
  Serial.println("!READ(Slider)");      // read the slider
  setPoint = Serial.parseInt();        // change the setPoint to reflect it

  sensorValue = analogRead(analogInPin);    // get pot voltage into sensorValue

  // Check pot value against setpoint, if above light LED
  // And take snapshot is sensorValue ? setPoint

  if (sensorValue > setPoint)
  {
   if (LEDstate == 0)           // take snapshot if above setpoint,
                          // just once each crossing (if LEDstate was off)
   {
    Serial.println("!O butSnap=1");    // Turn on Snapshot button
    Serial.println("!O butSnap.Run");  // Run snapshot event code
    Serial.println("!O butSnap=0");    // Turn off snapshot button
   }
   LEDstate = 1;
  }

  else
   LEDstate = 0;
  digitalWrite(LEDpin,LEDstate);

  // Check states of pushbuttons, if pressed change setpoint up or down
  SW1state = digitalRead(SW1pin);
  if (SW1state == 0)
  {
   setPoint++;                    // increment setPoint by 1
   Serial.print ("!O Slider =");    // send to Makerplot to adjust slider
   Serial.println(setPoint);       // based on new setPoint
  }

  SW2state = digitalRead(SW2pin);
  if (SW2state == 0)
  {
   setPoint--;                    // decrenemt setPoint by 1
   Serial.print ("!O Slider =");    // send to Makerplot to adjust slider
   Serial.println(setPoint);       // based on new setPoint
  }

  // print the analog values formatted for MakerPlot
  Serial.print(sensorValue);  // send 1st value
  Serial.print(",");         // send comma delimiter
  Serial.println(setPoint);   // send 2nd value with carriage return

  // print the digital values formatted for MakerPlot
  Serial.print("%");         // send binary indicator
  Serial.print(SW1state);    // send 1/0 for SW1
  Serial.print(SW2state);    // send 1/0 for SW2
  Serial.println(LEDstate);   // send 1/0 for LED with carriage return


  // wait 100 milliseconds before the next loop
  delay(100);
}
```

Figure 4. Part 2 of bi-directional setpoint sketch.

sketch that does the work just described. (This is really one sketch, but it's too large to fit into one box, so that's why it's divided into two sections.) As always, you can download and install this sketch by going to **www.makerplot.com** and clicking on the menu item **Arduino Sketch Examples ➔ MakerPlot Arduino Bi-Directional Control – Part 2.**

Let's look at the first part now.

Remotely Controlling MakerPlot

Figure 3 is the Setup part of the sketch, and where the constants and variables are defined. It then goes on to use MakerPlot instructions to configure the **Interactive Interface** to change the horizontal time axis to two minutes and to display the **"Controlled from Arduino"** message. We did this in Part 5, so it's the same.

What's new is the code that turns the four LEDs and toggle switches ON and then turns them OFF again. We've included this in the Setup part of the sketch to run only once in order to illustrate how your micro can send MakerPlot instructions via the serial channel to affect controls on the Interface. Here's what it does.

There are four LEDs and toggle switches on the top of the **Interactive Interface** (**Figure 2**). Normally, these controls would react to your mouse clicks by changing state but, instead, this part of the code does it from the Arduino. It's important to understand how it's done, since this is how to control MakerPlot from your micro.

Looking at the first group of instructions below, you can see how the **'!O'** prefix alerts MakerPlot to the fact that what follows next is an instruction. These three instructions simply turn on LED0 and SW0, and sound the bell audio tone. Then, it delays for a second. There are three more code groups that do the same thing for the other three LEDs and toggle switches.

```
Serial.println("!O LED0=1");
   // turn ON LED0 on the interface
Serial.println("!O SW0=1");
   // turn ON SW0 on the interface
Serial.println("!BELL");
   // sound PC Bell
delay(1000);
   // delay 1 second
```

While this group of instructions turned the LEDs and toggle switches ON one at a time, the next three lines of code turn OFF all the LEDs and toggle switches all at once:

```
Serial.println("!O LED*=0");
   // turn OFF all LEDs
Serial.println("!O SW*=0");
   // turn OFF all Switches
Serial.println("!BELL");
   // sound PC Bell
delay(1000);
   // delay 1 second
```

If you look closely, this is done with the "Wild Card" asterisk (*) symbol that tells all MakerPlot controls that have **"LED"** or **"SW"** as their labels to turn OFF (logic 0). So, although you can't physically see what happens to the LEDs and toggle switches in this article, you can go to our website and view the video at **Learn ➔ Arduino Bi-Directional Control – Part 2.** You'll also have the benefit of seeing first hand what happens to the rest of the setpoint code that is described next.

Sliding into Home

Now, let's show you how our new setpoint adjustment using the MakerPlot **Slider** control and the Arduino's SW1 and SW2 pushbutton switches work. **Figure 4** is the *loop* part of the sketch that does it; this is how it begins:

```
void loop() {
  Serial.println("!READ(Slider)");
   // read the slider
  setPoint = Serial.parseInt();
   // change the setPoint to reflect it
   // get pot voltage into sensorValue
  sensorValue = analogRead(analogInPin);
```

The first instruction sent to MakerPlot is the "**!READ (Slider).**" The Slider control on the **Interactive Interface** goes from 0 to 1000, so this instruction pulls this value into the Arduino's UART buffer. Following this, the next (Arduino) instruction pulls this value out of the UART buffer and places it in the **SetPoint** variable using the Arduino **Serial.parseInt()** instruction. So, this is how we do Steps 1, 2, and 3 in **Figure 1**.

The last instruction in this section simply reads the potentiometer value (0 to 1023) into the **SensorValue** variable, which we'll eventually compare against the **SetPoint**. So, with these three lines of code we now have the **Slider** value from MakerPlot, as well as the potentiometer value in separate variables. We could have stopped there, but we also wanted to get the two pushbutton switches involved in order to fine-tune the **SetPoint** value. Here's how that is done.

Fine-Tuning Things

Skipping over the part of the code that snaps an image of the Interface if the setpoint value is exceeded, let's proceed to the part that adjusts the **SetPoint** variable:

```
// Check states of pushbuttons, if
// pressed change setpoint up or down

SW1state = digitalRead(SW1pin);
   // get SW1 state

if (SW1state == 0)
   // test for SW1 depressed
{
  setPoint++;
   // increment setPoint by 1
  Serial.print ("!O Slider =");
   // send to Makerplot to adjust
   // slider
  Serial.println(setPoint);
   // based on new setPoint
}

  SW2state = digitalRead(SW2pin);
   // get SW2 state

if (SW2state == 0)
   // test for SW2 depressed
{
  setPoint—;
   // decrenemt setPoint by 1
  Serial.print ("!O Slider =");
   // send to Makerplot to adjust
   // slider
  Serial.println(setPoint);
   // based on new setPoint
}
```

The code is pretty straightforward:

- Read the switches (SW1 then SW2).
- See if either are depressed.
- If so, increment (SW1) or decrement (SW2) the **SetPoint** variable.
- Send the new setpoint to MakerPlot via the "**!O Slider = SetPoint**" command.

This part of the code satisfies Steps 4 and 5 in **Figure 1**. What follows is just sending the pot value along with the pushbutton switch settings to MakerPlot for plotting. Then, there's the 100 millisecond delay and everything repeats.

Figure 5 is just one screenshot of what it looks like.

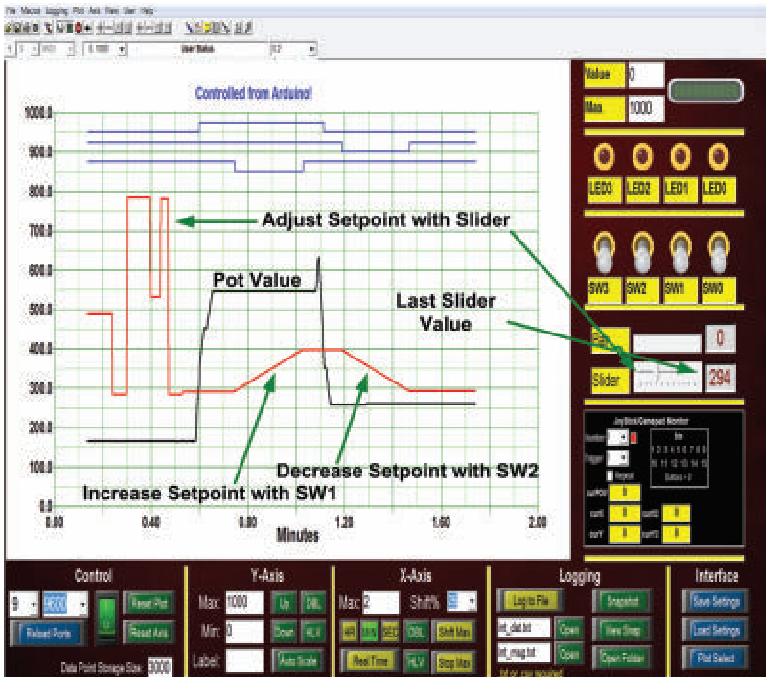


Figure 5. Bi-directional control using slider and pushbutton switches.

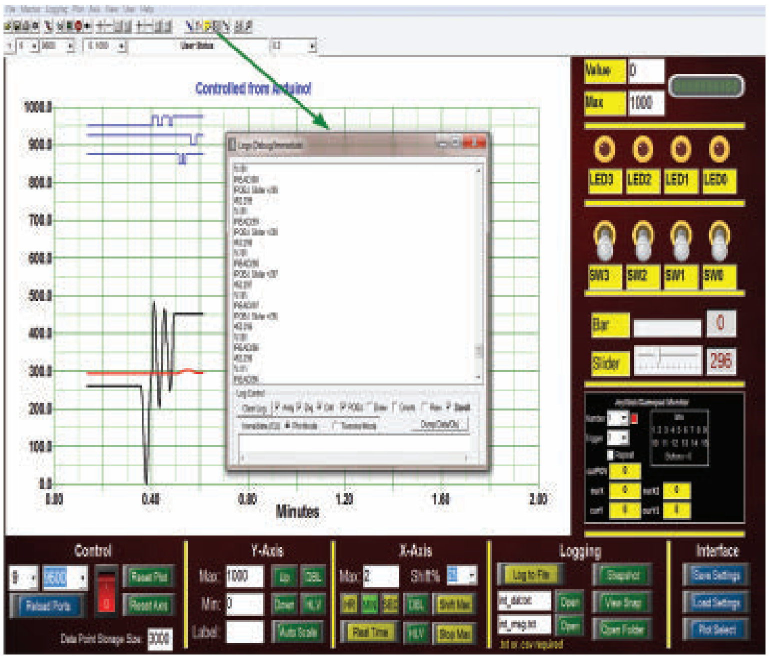


Figure 6. Behind the scenes with the Logs (Debug) Immediate window.

Behind the Scenes

Now that you've been shown how to code the Arduino to allow MakerPlot to monitor and control the setpoint, there's another MakerPlot feature we want to introduce you to: the **Logs(Debug) Immediate** window. The **Logs(Debug) Immediate** window can display a complete record of all data and other activities in MakerPlot. It is *the* primary MakerPlot debug tool!

You can bring up the **Logs(Debug) Immediate** window by clicking on the Logbook icon on the toolbar. It's shown in **Figure 6** where it's displaying the last part of the setpoint adjustment. Here, you can see not only the analog and digital data coming in from the Arduino, but also the **READ** values of the **Slider** control that MakerPlot sent back. That's followed by the **!POBJ Slider = xxx** values that the Arduino sketch sends back to MakerPlot to adjust the **Slider** control.

Without going into too much detail, the take-away from this is that with the **Logs(Debug) Immediate** window you have a complete record of what MakerPlot "sees" as data, and instructions are recorded both coming and going.

The **Logs(Debug) Immediate** window is invaluable in not only debugging general data but also in learning how MakerPlot works behind the scenes. As a matter of fact, that's how this series of articles began by showing you how MakerPlot can be a great debugging tool for your micro's code.

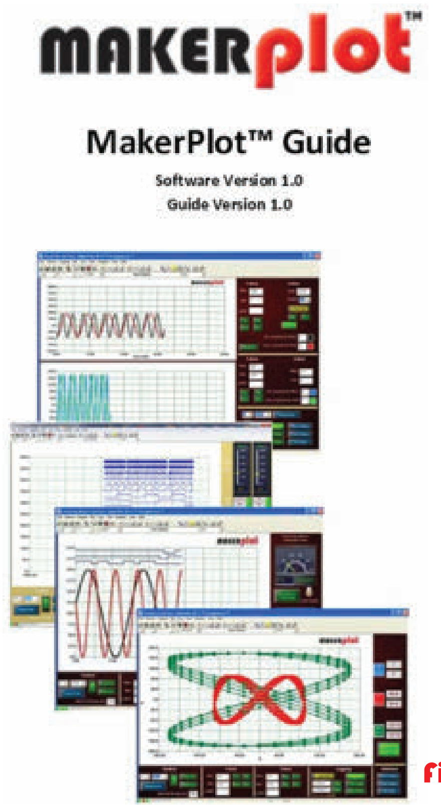
Conclusion

So, to sum up our bi-directional example to this point, we showed you how you can use MakerPlot as a front panel GUI to monitor and control your micro's code. In effect, anything you see on the MakerPlot Interface is accessible and controllable from your micro — whether it's an Arduino or whatever. We could go on with much more about bi-directional control, but that will be left to your imagination.

Remember, MakerPlot is all about customization and interactive control, so we're sure you'll think up ways to make it interact with your particular application the way you want it to.

For the next article, we're going to get more into the **Logs(Debug) Immediate** window capabilities as it will help you immensely in understanding how MakerPlot works. In the meantime, remember that MakerPlot is available as a free trial download. If you like what you see and what it does, you can order it from the *NV* Webstore at a discounted price.

That's all for now, so just remember: ***Got Data – MakerPlot It!*** **NV**



MAKERplot™

MakerPlot™ Guide

Software Version 1.0

Guide Version 1.0

Figure A. MakerPlot Guide.

MakerPlot Guide

This article introduces some instructions that allow you to interact with MakerPlot from your micro. However, there's a lot more to programming MakerPlot than that. There are also instructions for building all the graphic controls and placing them on the Interface screen, plus ways to use MakerPlot as a math coprocessor for integer-based micros. All of the MakerPlot instructions can be found in the MakerPlot Guide at **www.makerplot.com**.

Beginner's Guide Complete Combo!

Beginner's Guide to Embedded C Programming
VOL 3 Experiment Component Pack

Combo Price $229.95

For complete details, visit our webstore @ www.nutsvolts.com.

Call to order at 1-800-783-4624 or go to www.nutsvolts.com

ALL ELECTRONICS

www.allelectronics.com

Order Toll Free 1-800-826-5432

16.5MM ENCLOSED PIEZO ELEMENT

East Electronics# TFM-23.
80dB 4kHz resonant frequency.
4" pigtail leads.
CAT# PE-59

75¢ each

12V BLUE LED, 5MM

Bright 5mm blue LED (milky-white in off-state) with built-in resistor for 12 Vdc, No external resistor required. Works well on 4-12Vdc. Dimmer at lower voltages. Diffused. 35° viewing angle.
CAT# LED-12BW

55¢ each

100 for 45¢ each
1000 for 35¢ each

2.5MM COAX PLUG ON 4" CABLE

20 gauge, stranded zip cord.
CAT# CB-238

$1.85 each

10 for $1.70 each

WALL-MOUNTABLE TUBE CLAMP, USED

Heavy-duty aluminum saddle for up to 1" diameter pipes. 63 x 75mm aluminum wall plate with four recessed mounting holes. Clamp can be positioned horizontally or vertically on plate or used separately from plate. Removed from used equipment. **CAT# CLMP-3**

$3.00 each

230 UF 330 V PHOTO-FLASH CAPACITOR

Rubycon. 15mm dia. x 43mm.
Leads on 7mm centers.
CAT# PPC-233

$1.35 each

100 for $1.20 each
500 for $1.00 each

5VDC "MINI-CUBE" SPDT RELAY

Midtex # MMS-105.
5 Vdc, 75 Ohm coil. SPDT 2 Amp contacts. 10 x 10 x 11mm. PC leads.
CAT# RLY-567

70¢ each

10 for 60¢ each
100 for 45¢ each

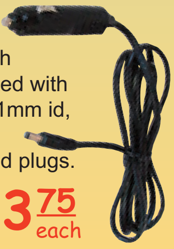
NEODYMIUM MAGNET, 28 X 16 X 4.5MM

Very strong for its size, 18-22K gauss.
Irregularly shaped.
1.10" x 0.63" x 0.19".
CAT# MAG-148

Reduced Price! $1.25 each

10 for $10.00

6' CIGARETTE LIGHTER POWER CORD



Automobile power plug with replaceable fuse. Terminated with co-ax power connector, 2.1mm id, 5.5mm od, 12mm length.
Very good quality cable and plugs.
CAT# CLP-300

$3.75 each

10 for $3.50 each

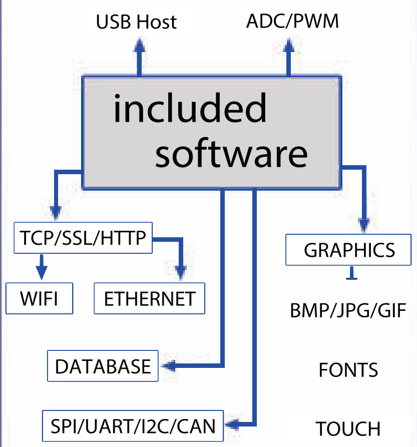
PANEL METER, FULL SCALE-1MA

2.4" square panel meter.
Full scale = 1mA DC.
Imprinted with 0 - 15 and 0 - 3 VOLTS scales in black with white background.
CAT# PM-376

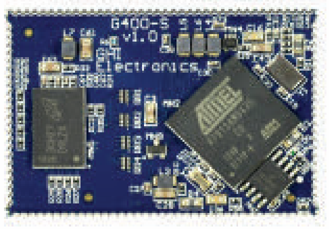
$3.50 each

Program Using

.NET C# VB

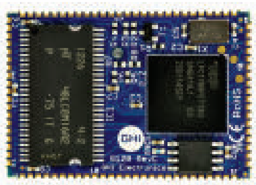


G400
System on Module



400MHz ARM 9
128MB RAM
$60 in volume

G120
System on Module



• 120MHz Cortex M3
16MB RAM
$30 in volume

RoHS FC CE

GHIElectronics.com

BY JOE PARDUE

The Arduino Classroom
Arduino 101 — Chapter 3: How an Arduino Program Works

This article continues the Arduino 101 series as part of a formal curriculum where you learn computing and electronics basics much like you'd learn if you took a standard semester-based introductory course. To help in this, I've started www.arduinoclassroom.com where each of these magazine articles will be presented along with laboratories, exercises, quizzes, and a forum. Now, instead of being a passive reader, you get a chance to have some interaction with other readers and to talk back to the writer. This series uses an Arduino proto shield and some electronic components that are available in the Arduino 101 Projects kit that you can get from the *Nuts & Volts* Webstore.

In the January and February articles, you were introduced to the Arduino, how to use it to blink an LED, and how to use a breadboard with it to build and test electronic circuits. This month, you'll be introduced to some basic concepts of an Arduino program and how to use those concepts to keep counts and times so that we can blink LEDs in programmable patterns.

Program Versus Sketch

The Arduino community calls a **program** a **sketch**. This is because the Arduino was originally written for artists, but nonetheless, a sketch is a computer program. We will use both terms without too much regard to formality. When discussing concepts general to computers, we'll probably use 'program' and when we are referring to a specific Arduino program, we might use 'sketch.' So, be prepared to see it either way.

We define a program (sketch) as a sequence of human readable text statements following the rules of a **programming language**. This program can be converted to **instructions** that a computer can follow to do the sorts of things a computer does.

A computer program is converted to a sequence of instructions that a computer understands, then those instructions can be **uploaded** to the computer to make it do what we tell it to do in the program. These programs are converted from human readable text to machine readable code by a **compiler** which is also a computer program. The process is called **compiling**, though the Arduino community calls it **verifying** (or **verify**). We'll also use these terms both ways.

How Do We Write a Program?

Let's repeat the above: A program is some text that follows some rules and can be converted into something that we can feed the computer to use to make decisions about what it is to do next. Text, rules, and decisions — we learn the rules, write the text, then the computer makes decisions and does something. This is the sequence:

1. We decide what we want the computer to do.
2. We think about how our programming language rules might be used to get the computer to do our bidding.
3. We write a program using the text editor in the Arduino IDE (Integrated Development Environment).
4. The Arduino IDE sends that text to a compiler on our PC that checks our text against some rules and then builds something that can be uploaded to the computer on the Arduino board. The computer on the Arduino accepts the uploaded data and starts making decisions about how the Arduino board processes information and uses the hardware.

Following this list means that in order for us to write a program we are going to have to learn two things: What are the rules we can use and what are the decisions the computer can make? Fortunately, we don't have to learn it all at once before we get started playing with them. We will learn bits and pieces in small steps, playing around as much as we like until it starts making sense.

How an Arduino Program is Structured

An Arduino program is structured in four parts:

FIRST: Begin with some comments about the program.
SECOND: List **variables** and **constants** that all the functions may use. Variables are names for memory locations that a computer can use to store information that might change. Constants are numbers that won't change.
THIRD: Run the *setup()* function to get the system ready to run the *loop()* function. This is where you perform tasks that you want done once at the beginning of your program:

```
void setup()
{
  // do things once at the start of the program
}
```

(SIDE NOTE: The '//' tells the compiler that the line is a comment and that it isn't code that needs to be compiled.)
FOURTH: Run the *loop()* function. This is where you run things in a sequence from the top of the loop to the bottom of the loop, and then you start over again at the top, looping until the machine gets turned off:

```
void loop()
{
  // Do the first thing

  // Do the second thing

  // Do any number of things

  // Do the last thing in the list

  // Go back to the beginning of this list
}
```

Structure of the Arduino Blink Example

I have taken the Blink program from the Arduino Examples and added the lines showing each of the four sections:

FIRST: Opening comments.

```
/*
 * Blink
 *
 * The basic Arduino example.  Turns on an LED
 * on for one second, then off for one second,
 * and so on...  We use pin 13 because,
 * depending on your Arduino board, it has
 * either a built-in LED or a built-in resistor
 * so that you need only an LED.
 *
 * http://www.arduino.cc/en/Tutorial/Blink
 */
```

SECOND: List variables.

```
int ledPin = 13; // LED connected to digital pin
```

THIRD: Things to do once at the beginning.

```
void setup() // run once, when the sketch starts{
  pinMode(ledPin, OUTPUT);
              // sets the digital pin as output
}
```

FOURTH: Do things in a loop.

```
void loop()             // run over and over again
{
  digitalWrite(ledPin, HIGH);
    // sets the LED on
  delay(1000);          // waits for a second
  digitalWrite(ledPin, LOW);
    // sets the LED off
  delay(1000);          // waits for a second
}
```

In the January (Chapter 1) article, we saw how to take the example program Blink, load it in the Arduino IDE, compile it, and upload it to the Arduino board. You can locate this program in the Arduino IDE menu at File\Examples\Digital\Blink. Now would be a good time to repeat that lesson. Click on the Blink example, then follow the instructions from Chapter 1 to compile and upload.

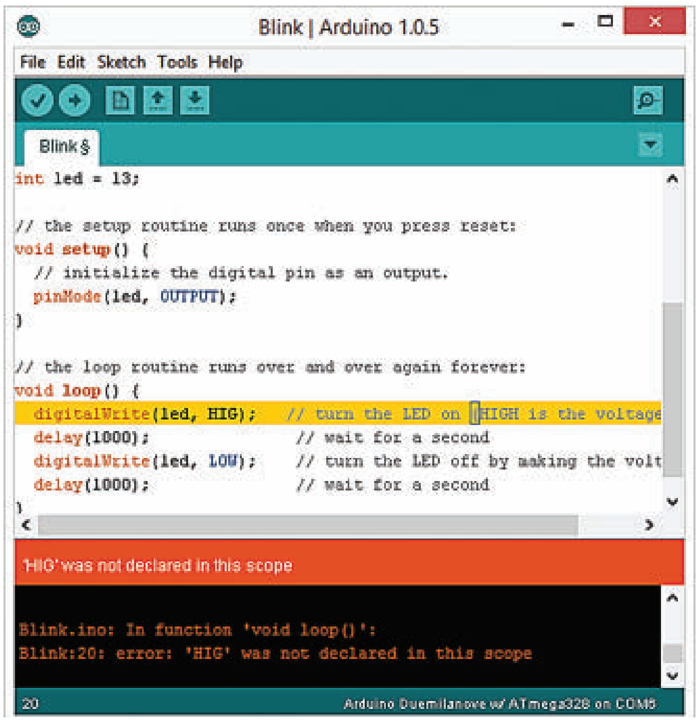
What if It Doesn't Compile?

We write a software program using tools that allow us to create these computer instructions in some human comprehensible form, like text files that use words and symbols available on a keyboard. Our text file (**source code**) must follow a set of rules that are provided for a specific programming language, and then we must submit that file to some compiler software that will then do one of two things: compile it or refuse it and give you hints about what went wrong that provide a guide to making the source file more palatable to the compiler.

We will intentionally make a mistake in the Blink example to generate an error by deleting the last H in the parameter HIGH in the *digitalWrite* function. The compiler can find HIGH but it can't find HIG, so it stops the compilation and generates an error as shown in **Figure 1**.

The process of finding errors in a program is called **debugging**. In the example given, debugging is easy since the Arduino IDE tells you what the problem is and even shows you where it is located. The error message was: *"'HIG' was not declared in this scope"* and the line with the error was highlighted in orange.

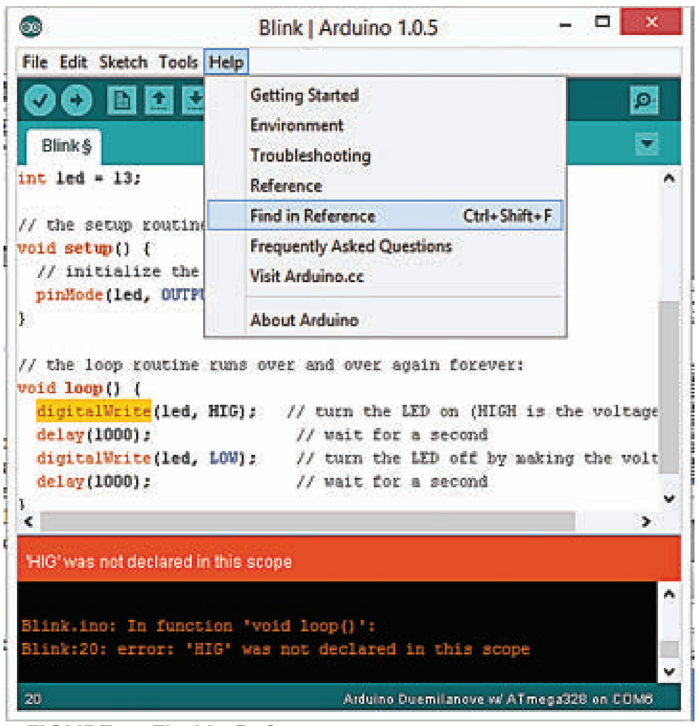
Post comments on this article and find any associated files and/or downloads at
www.nutsvolts.com/index.php?/magazine/article/march2014_Smiley.



```
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIG);   // turn the LED on (HIGH is the voltage
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the volt
  delay(1000);               // wait for a second
}
```

```
'HIG' was not declared in this scope

Blink.ino: In function 'void loop()':
Blink:20: error: 'HIG' was not declared in this scope

20                    Arduino Duemilanove w/ ATmega328 on COM6
```

FIGURE 1: *Error.*



```
int led = 13;

// the setup routin
void setup() {
  // initialize the
  pinMode(led, OUTPU
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIG);   // turn the LED on (HIGH is the voltage
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the volt
  delay(1000);               // wait for a second
}
```

```
'HIG' was not declared in this scope

Blink.ino: In function 'void loop()':
Blink:20: error: 'HIG' was not declared in this scope

20                    Arduino Duemilanove w/ ATmega328 on COM6
```

FIGURE 2: *Find in Reference.*

The message may be somewhat cryptic at this point in your learning since we haven't discussed what 'declared' or 'scope' mean, but by pointing out that HIG was giving the compiler problems, you can easily guess that it is the culprit. If you don't already know that the only valid second parameters for the *pinMode* function are HIGH or LOW, you can highlight the *digitalWrite* function as shown in **Figure 2** and then open the Help menu and click on Find in Reference.

This will open the page for *digitalWrite* where — as you can see in **Figure 3** — the values for the second parameter are HIGH and LOW.

If we add back the H, then the program will compile. While this example was simple, program debugging often isn't so easy since the error messages may be quite cryptic. When it isn't entirely clear what is going on, it's probably best to refer to code that you know works and has a similar line in it so you can see what you might be doing differently. If you still can't figure it out, then it is time to go on the Internet and ask the question on a forum (more on that process later). We have been using some terms that we haven't yet defined (like *function*, *int*, and *void*). We'll do that a lot in our study of the Arduino since there is just too much to introduce in a perfect order all at once. We'll get to those terms below.

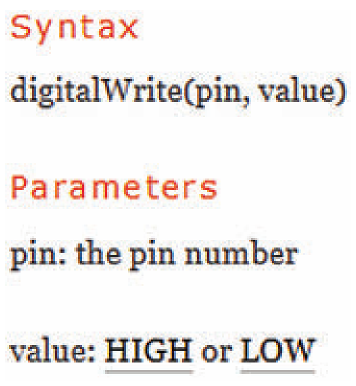
Learning the Programming Language Rules and Concepts

Arduino is — among other things — a programming language. One of the first things to learn about the Arduino language is that underlying it is the C and C++ programming styles. What Arduino does is provide a novice-friendly buffer that hides some standard C and C++ requirements, and adds some pre-written functions that help the novice use the Arduino board. [They also provide many libraries of functions and many working projects to use as examples.] You don't need to know a thing about C or C++ in order to use Arduino or follow this series.

Since the Arduino keeps it simple, you can often just look at existing code to get an intuitive feel for what is going on and pick up a lot of the rules by osmosis. Of course, as you grow in this process and want to do more complex things, you'll also want to learn more about the rules. Let's first introduce some rules for a programming concept called the variable.

What is a variable?

As we learned earlier in this chapter, variables are programming elements that can take different values. When we define variables, we must tell the compiler what **type** of a variable it is. To a compiler, the variable type indicates the



Syntax

digitalWrite(pin, value)

Parameters

pin: the pin number

value: HIGH or LOW

FIGURE 3: *Found in Reference.*

amount of data that must be stored. For instance, a variable with a **char** type can store any number from 0 to 255, and an **int** type can store any number from 0 to 65535. In C, we define our variables either at the beginning of a code **module** (usually a file) outside of any one function for it to be available to any function in that module, or we define the variable within a function for it to be available only to statements within that function. For example:

```
// Module start

char myChar;

void myFunction(){
       int myInt;
       // statements
}

// Module end
```

This example defines the **char** type variable *myChar* that can be used by any function in the module, and the **int** type variable *myInt* that can be used only by statements in the *myFunction* function.

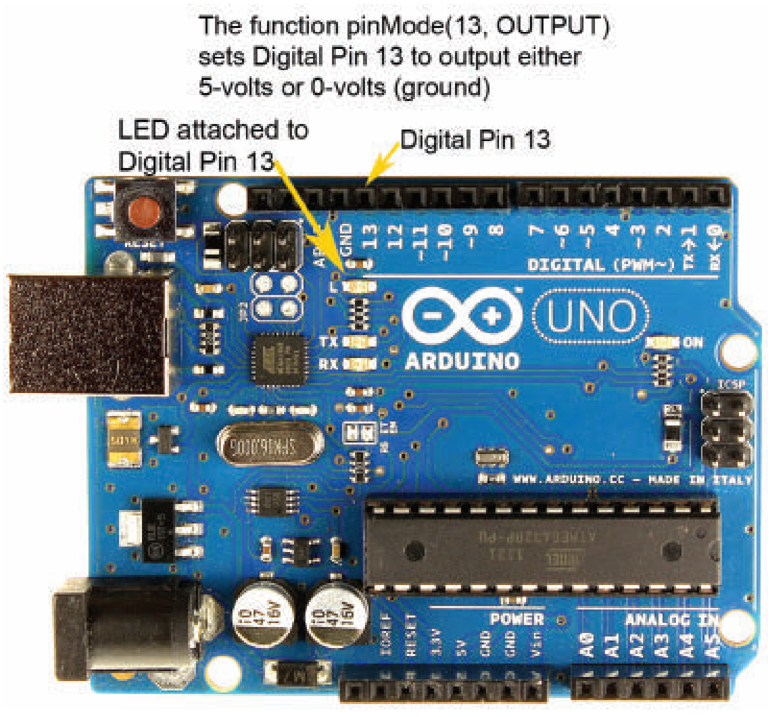

The function pinMode(13, OUTPUT) sets Digital Pin 13 to output either 5-volts or 0-volts (ground)
LED attached to Digital Pin 13
Digital Pin 13

FIGURE 4: *pinMode.*

What is a function?

We have been using the word **function** which has a special meaning in our programming. It refers to a block of software that can be used by referring to the function name, followed by parentheses containing a list of information that can be used by the function. For the Arduino, an example is the line shown in **Figure 4** with the function *pinMode(13, OUTPUT)*. The function name is 'pinMode' and it will use the information given between the parentheses (which in this case is the number 13 and the word OUTPUT, separated by a comma).

When the program uses the *pinMode(13, OUTPUT)* function, the computer will call a block of software from a library of Arduino functions. This unseen-by-you software will use the 13 and OUTPUT to set the Arduino pin 13 to output voltage signals, meaning the program can set the voltage on pin 13 to five volts or zero volts (ground). Since this pin is also connected to an LED on the Arduino, setting the *pinMode* to output lets us provide the LED with +5 volts to turn it on, or set it to zero volts (ground) to turn the LED off. After setting the mode for the pin, we can then use the *digitalWrite* function to control the voltage. By writing the *digitalWrite(13,HIGH);* statement in your program, the Arduino will provide five volts to the LED shown in **Figure 1** and cause it to light up. Adding the statement *digitalWrite(13,LOW);* causes it to set the voltage to zero, thus turning the LED off.

How Does an Arduino Program Flow?

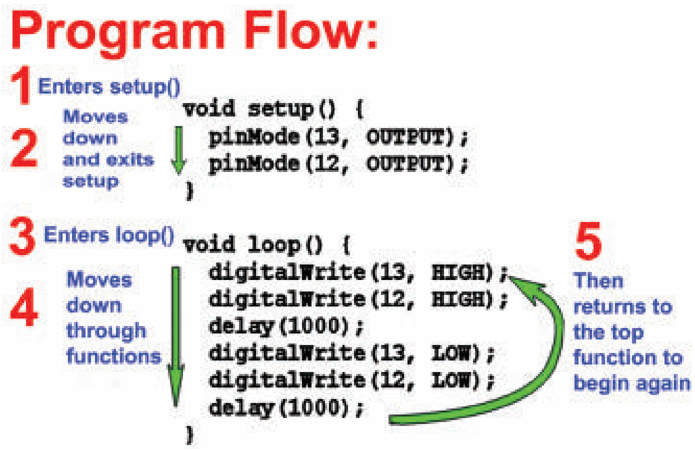
We already looked a bit at how a program works, but let's look again from a different angle (we'll use a picture this time) to reinforce the important concept of program flow. An Arduino program has a minimum of two functions: *setup()* and *loop()*. Neither of these functions take parameters in the parentheses. As shown in **Figure 5**, the *setup()* function runs once and is used to set up the Arduino. The second function *loop()* will run each function in a loop: a sequence beginning at the top. When it has run the last function, it will go back to the top of the list and run through them again, looping through them continuously.

In **Figure 5**, we use *pinMode* to set up pins 12 and 13 as outputs, and that we use the library functions *digitalWrite* and *delay* in the loop to put five volts on each pin, wait a second, put zero volts on each pin, wait a second, and then go back to the top of the loop to repeat the sequence until the Arduino is turned off.

We saw previously how Arduino software flows for a program that does something very simple. Most programs do things that are much more complex, and programming languages provide tools for controlling the flow of the program (controlling the execution sequence of the functions). For example, you might want to control the flow of a program so that the LED turns on and off once a second forever as we've seen with the Blink example. Or, you may want the program to only turn the LED on and off 10 times. Let's look at some of the core concepts involved.

The 'while' Flow Control Statement

In the Blink program from the Arduino Examples



```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  digitalWrite(12, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  digitalWrite(12, LOW);
  delay(1000);
}
```

Program Flow:
1 Enters setup()
2 Moves down and exits setup
3 Enters loop()
4 Moves down through functions
5 Then returns to the top function to begin again

FIGURE 5: *Program flow.*

directory, we saw how to blink an LED in one second intervals forever. What if we want to blink the LED five times and then stop blinking? For this, we will need to use a variable to contain the data for the number of times to blink. We will also use the C control flow statement **while** to loop through a block of code and repeat the action of that block the number of times indicated by the variable. This is easier to understand by looking at the code rather than talking about it, so:

```
void loop() {
  // define the count variable and set it to 5
  char myCount = 5;

  // run the following while count is greater
  // than 0
  while(myCount){
    digitalWrite(13, HIGH); // set the LED on
    delay(1000);            // wait for a second
    digitalWrite(13, LOW);  // set the LED off
    delay(1000);            // wait for a second
    myCount = myCount - 1;  // decrement the
                            // count
}
```

We start the *loop* function by defining a variable *myCount* with a type of *char*, and we set the value of that variable to five using the **equate operator** '=.' Next, we create a block of code that looks like a function but isn't; we write *while(count){*, then the code to run while *count* is **true** — meaning that *count* is not equal to zero.

The *while(count)* block begins with an **open brace** '{' and ends with a **close brace** '}' which is the way we create a block of code in C. It doesn't matter much to the *while* control flow statement; the C code will run whatever is in the block and then return to the *while(expression)* and check it to see if it's still true. If it is, it runs the block again and then rechecks, repeating the process until the expression becomes not true (equal to zero).

In our example, the expression is *myCount* and it starts out as five. At the end of the *while(myCount)* block, the count is **decremented**, meaning that the value of *myCount* is decreased by one so that the first time through *count* becomes four. Then, when the program returns to the *while(myCount)* statement, it sees that *myCount* is four which isn't zero, so it repeats the block making *myCount* equal three. It repeats for two, one, and zero. When the program returns to the *while(myCount)*, this time it sees that *myCount* is zero which is by definition **false**. So, the program exits the *while* loop.

One interesting thing about the *while* control flow statement is that it can be used to stop a program and have it spin its wheels forever using:

```
while(1);
```

The C program evaluates the expression and sees that it is true, so it does what the statement tells it to do between the closing parenthesis and the semicolon — which is nothing. It then reruns the statement and examines the *while(expression)* statement and sees that it is true. You get the picture? Yes, there are times you want to do this — like when you want to have a program stop. In Lab 1, we will use *while* to blink the LED five times, then do nothing after that.

The 'for' Flow Control Statement

We could have accomplished the same task (blink the LED on and off five times) using the **for** control flow statement. The rules for this statement can get somewhat complex, but we will only use it in the form that lets us do a block of code for a specific number of counts:

```
for(count = 0; myCount < 5; myCount++){// run
the following 5 times
```

This statement uses three math operators used in three statements within the parentheses that follows *for*: the '=' (equate operator); the '<' (less-than operator); and the '++' (post-increment operator). We've seen that = sets the variable on the left of the operator to the value of the variable or constant on the right. For the *myCount < 5* case, it is the constant 5. The < operator lets the second statement *count < 5* evaluate if *myCount* is, in fact, less than five. It asks, 'Is it true that *myCount* is less than 5?' If that is true, then the following block of code (code between the { and }) is allowed to run. Before the block of code is run, however, the third statement in the parentheses adds one to *myCount*.

So, the first time the statement is run by the program, the count is set to zero in the first statement; the truth of the statement *count < 5* is judged (it is true); then in the third statement, *myCount* is incremented so that it becomes equal to one. Now, the block of code is run by the program to blink the LEDs and the program returns to the *for* statement. However, this time, it doesn't set *myCount* to zero in the first statement. That is only done for the first pass through the block.

It checks the second statement and sees that it is true that *myCount* (which is now one) is less than five. In the third statement, it increments *myCount* so that it now equals two, and it runs the block of code blinking the LED again. This continues until *myCount* has been incremented to equal five and the *myCount* < *5*; statement is false. The program doesn't run the block again, but jumps over it and runs whatever statements follow. [In our case, this is *for(;;)* and as with *while(1)*, the *(;;)* is always true so the program stops blinking the LED and spins its wheels forever.]

```
void loop() {
  int i;

  for(i = 0; i < 5; i ++){// run the following
  5 times
    digitalWrite(13, HIGH); // set the LED on
    delay(250);             // wait for a second
    digitalWrite(13, LOW);  // set the LED off
    delay(250);             // wait for a second
  }

  for(;;); // Spin your wheels forever
}
```

You might wonder why there are two different ways to do the same thing — *while* loops and *for* loops. The answer is that there are more advanced cases that we will see later, where either *while* or *for* has an advantage. In Lab 2, we will use *for* to blink the LED five times, then do nothing after that. Now, let's take some time to do some hands-on lab exercises to reinforce and expand on what we've learned so far.

Lab 1: Counting.

Parts required:

1 Arduino
1 Proto shield

Estimated time for this lab: 10 minutes

Check off when complete:

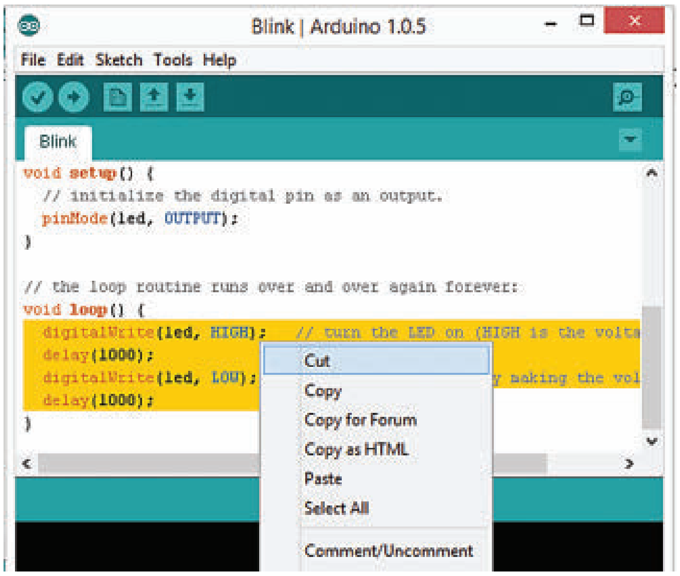
❑ Open the Blink program.
❑ Delete the content of the *loop()* function by highlighting the code with the mouse cursor right button pressed. Then, press the left mouse button and select 'Cut' as shown in **Figure 6**.
❑ Type into the *loop()* function the code shown below and in **Figure 7**:

```
int i;

for(i = 0; i < 5; i ++){// run the following
5 times
  digitalWrite(13, HIGH);   // set the LED on
  delay(250);             // wait for a second
  digitalWrite(13, LOW);   // set the LED off
  delay(250);              // wait for a second
}

for(;;); // Spin your wheels forever
```

❑ Compile and upload the program and verify that the LED blinks five times and stops.
❑ Beneath the *int i*; statement, write *char myCount* = *10.*
❑ In the *loop()* function *for* statement, substitute *myCount* for the 5 as shown below:

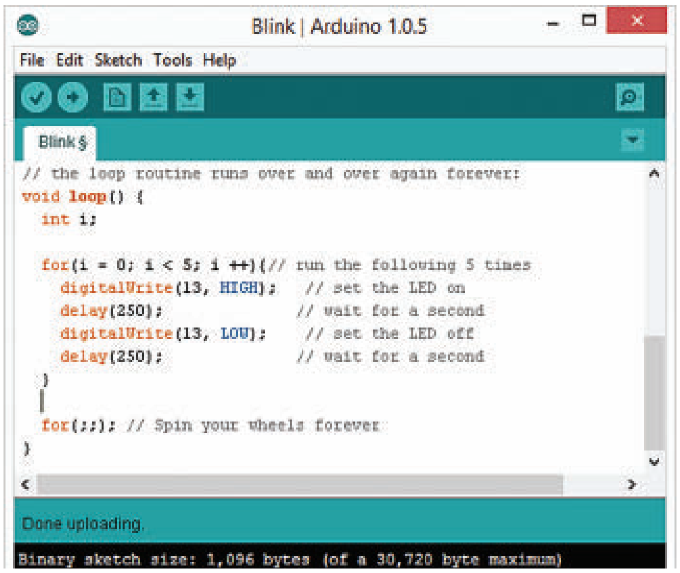
```
int i;
char myCount = 10;


for(i = 0; i < myCount; i ++){// run the
following 5 times
```

❑ Compile and upload the program, and verify that the LED blinks 10 times and stops.



```
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the volta
  delay(1000);
  digitalWrite(led, LOW);                            y making the vol
  delay(1000);
}
```

■ FIGURE 6: *Delete contents of loop.*



```
// the loop routine runs over and over again forever:
void loop() {
  int i;

  for(i = 0; i < 5; i ++){// run the following 5 times
    digitalWrite(13, HIGH);   // set the LED on
    delay(250);              // wait for a second
    digitalWrite(13, LOW);    // set the LED off
    delay(250);              // wait for a second
  }

  for(;;); // Spin your wheels forever
}
```

```
Done uploading.
Binary sketch size: 1,096 bytes (of a 30,720 byte maximum)
```

■ FIGURE 7: *Blink modified to count.*

Lab 2: Timing.

We have used the *delay(250)* function in several of our examples. The 250 is for 250 **ms** (**milliseconds**) [there are 1,000 ms in a second, so 250 ms is a quarter of a second], but it can be any number of ms up to the maximum allowed for the *delay* parameter which is that of an **'unsigned long.'** For the Arduino, an unsigned long can have values from 0 to 4,294,967,295 — that is a lot of milliseconds. In this lab, we will see how to use the *delay* function to time the blinking of an LED.

Parts Required:
1 Arduino
1 Mini breadboard shield

Part 1 — Blink Timing

Sometimes we want to control the timing of events. In the original Blink program, we controlled the LED on and off times with the *delay()* function by setting the number of milliseconds to 1,000 (one second). What happens if we shorten that interval?

Estimated time for this lab: Five minutes

Check off when complete:
❏ Open the Blink example program and change the 1,000 to 500.
❏ Run the program and note that the LEDs are blinking faster.

If we continue to reduce the number of milliseconds, the LED will appear to blink faster and faster. At some point while lowering that delay value, it will blink so fast that we can no longer perceive that it is blinking. Our eye tells us that the LED is on continuously. This phenomenon is called **Persistence of Vision** (**POV**). Let's do an experiment to see how fast we need to blink an LED to make our eyes think it is on continuously.

Estimated time for this lab: 15 minutes

Check off when complete:
❏ Open the Blink program and change the 1,000 to 100.
❏ Run the program and note that the LEDs are blinking faster.
❏ Change the milliseconds from 1,000 to 1.
❏ Run the program and note that the LED appears to be on continuously.
❏ Bracket the millisecond value until you find the highest value that causes the LED to appear to be on continuously to you.
❏ If you are working in a group, compare this value to the value other students have found.

Part 2 — Dimming the LED

You can turn the LEDs on and off at different intervals. For instance, you could turn them on for 100 milliseconds and off for 1,000 milliseconds. This would make the LED appear to flash on momentarily, then be off for a second. You can use this and POV to control the perceived brightness of the LED.

Estimated time for this lab: 15 minutes

Check off when complete:
❏ Set the *delay()* off time to the value you determined in the POV experiment. (In my case, this was 13 milliseconds).
❏ Set the *delay()* on time to one millisecond, then verify and upload the program.
❏ Note that the LED appears dimmer than in the original experiment.
❏ Set the *delay()* on time to half of your original POV value and notice that the LED appears brighter than the 1 ms blink rate.
❏ Play with the on and off times, and observe the effects.

What we're observing is that the LED is being turned on and off too fast to see it blink, but we're leaving it off longer than it's on so less total light is being seen.

Part 3 — Emergency Flasher

The example used in this lab comes from a contribution (thanks Rvw) to the **arduinoclassroom.com** forum posted at **http://arduinoclassroom.com/index.php/forum/chapter-1/19-tweaked-myblinky**.

Estimated time for this lab: 15 minutes

Check off when complete:
❏ Open the Arduino IDE.
❏ Delete the contents of the *loop()* and type in the following code:

```
/*
MyBlink3
Turn on an LED on for .05 seconds, then off for
.05 seconds three times in a row and then pauses
for one second. This gives an emergency flasher
effect.
*/

// Pin 13 has an LED connected on most Arduino
// boards. Give it a name:
int led = 13;

// the setup routine runs once when you press
// reset.
void setup() {

  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
```

```
// the loop routine runs over and over again
// forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on
  delay(50); // wait for .05 seconds
  digitalWrite(led, LOW); // turn the LED
  delay(50); // wait for .05 seconds
  digitalWrite(led, HIGH);
  delay(50);
  digitalWrite(led, LOW);
  delay(50);
  digitalWrite(led, HIGH);
  delay(50);
  digitalWrite(led, LOW);
  delay(1000); // wait for one second
}
```

❑ Play around with the delays and LED sequences to see how your changes affect the visual output.

Lab 3: Two LEDs – Railroad Crossing.

You may have seen how the lights at some railroad crossings blink in opposition to each other: One is on while the other is off. We can do this with two LEDs.

Parts required:

1 Arduino
1 Arduino proto shield
2 Red LEDs
2 1000 Ω resistors
2 Jumper wires

Check off when complete:

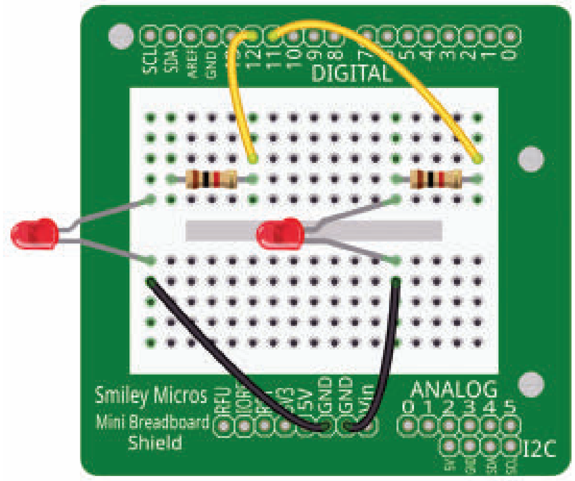
❑ Unplug your Arduino from the USB (and battery if one is used) to make sure the power is off before building the circuit.

❑ Follow the illustrations in **Figures 8** and **9**, and add the two LEDs and resistors to the breadboard.

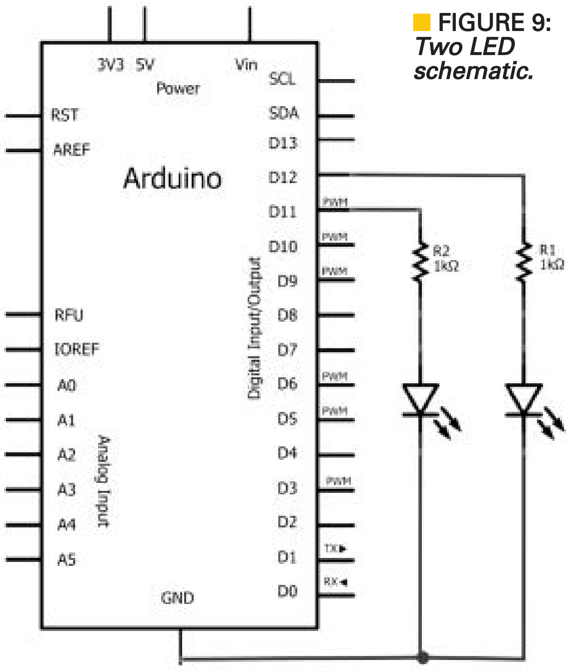
❑ Note that the LEDs have their long legs to the bottom of the resistors and the short legs are connected to ground.

❑ Based on what you learned about how to blink a single LED, spend some time to think about how you might blink two LEDs. Now, think about how you might blink two LEDs so that one is off while the other is on.

❑ Plug the USB cable into the Arduino.



■ FIGURE 8: *Two LEDs.*



■ FIGURE 9: *Two LED schematic.*

❑ Open the Blink example program and change it so that the LEDs on pins 12 and 11 blink in an alternating pattern at 1/2 second on and 1/2 second off.

You may very well have figured out how to do this on your own but whether you did or not, the following modification of the Blink program shows one way to accomplish the task:

```
void setup() {
  pinMode(12, OUTPUT);
    // initialize pin 12 as output
  pinMode(11, OUTPUT);
    // initialize pin 11 as output
}

void loop() {
  digitalWrite(12, HIGH);
   // set the LED on
  digitalWrite(11, LOW);
   // set the LED on
  delay(500);
// wait for 1/2 a second
  digitalWrite(12, LOW);
   // set the LED off
  digitalWrite(11, HIGH);
   // set the LED on
  delay(500);
   // wait for 1/2 a
   // second
}
```

This is a simple modification to what we have already learned. We only have to set one LED HIGH (turned on) and the other LED LOW (turned off).

Arduino Classroom Website and Forum

I'd like to thank all those who have signed into the **arduinoclassroom.com** forum to ask questions or just say hello. I think it's a real good supplement to the materials presented here. If you have questions, using the forum just might help.

Next Month

In Arduino 101 Chapter 4, we will learn some more Arduino software principles and begin learning about digital input by using pushbuttons for user interaction. **NV**

BY L. PAUL VERHAGE

A New Airframe Design for Near Spacecraft: Part 3 — Securing the Internals

After experiencing electronics failure during descent, I was determined to design an airframe where the near spacecraft avionics could never shift around in flight. In the past, I relied on tightly packing the interior volume with Styrofoam peanuts to keep things in place, but that led to the case of centripetal force knocking the stuffing out of the airframe. Now everything — including the batteries and GPS receiver — attaches to the airframe with bolts.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/march2014_NearSpace**.

So far, I've discussed the construction of a new near spacecraft's airframe and avionics pallet. I'll finish this series by explaining how the battery/GPS pallet and hatch are secured to the airframe. The new near spacecraft is a bit heavier than the older design but far less susceptible to failure due to the chaos of balloon burst and descent.

Battery/GPS Pallet

After securing the avionics pallet to the bottom of the airframe with bolts, I turned my attention to securing the flight battery. It's true that the avionics are more expensive than the flight battery. However, if post-burst chaos separates the battery from the avionics, I won't be getting the avionics back. This makes the less expensive battery just as important as the more expensive avionics.

The new design needed to take into consideration that I believe it's a bad idea to load batteries into the near spacecraft the night prior to launch. I'm a bit paranoid that somehow the power switch will accidentally be activated and the near spacecraft will drain its flight battery. Discovering a dead flight battery on the morning of launch is the fastest way to cancel the launch.

My policy is that the battery always remains disconnected and uninstalled until the crew begins filling the balloon. Therefore, it's necessary to place the battery into the top of the airframe rather than position it into the bottom where it's difficult to reach — especially on cold mornings.

Since my primary concern is preventing the contents of the airframe from separating from one another, I secure the flight battery to the airframe using a pallet — just like the avionics pallet.

While designing the battery pallet, I found that it was large enough to also secure the GPS receiver. The combined battery/GPS pallet attaches to the airframe with eye bolts. Bolting to the airframe means the battery/GPS pallet cannot come lose in the event of a hatch failure.

The pallet is constructed of three lightweight layers: two are permanently bolted together and a third acts as a lid. The bottom layer is Coroplast — a corrugated plastic made from polypropylene. This material is 1/4" thick and very rigid. It's available at some hobby stores and from most plastic suppliers.

Above the Coroplast is a sheet of foamed Styrene plastic (Styrofoam) and two polystyrene square tubes. I use 1/2" thick pink foam (house insulation) for the Styrofoam which I purchased from my local home improvement store. The polystyrene square tubes are from my local hobby shop.

The Styrofoam sheet attaches to the Coroplast with 1-1/2" long, 1/4"-20 nylon bolts and nuts, and the two polystyrene tubes attach to opposite ends of the Coroplast with #4-40 bolts and nylocks. Bolts are required to assemble the layers because adhesives do not stick to polypropylene Coroplast.

After bolting the bottom of the

pallet together, I cut pockets out of the Styrofoam to hold the flight battery and GPS receiver. There are also pockets for less important batteries like for the audio locator and servos.

I finished by cutting out channels for the cables connecting these devices to the avionics. The fit is tight so the batteries and GPS receiver can't shift around.

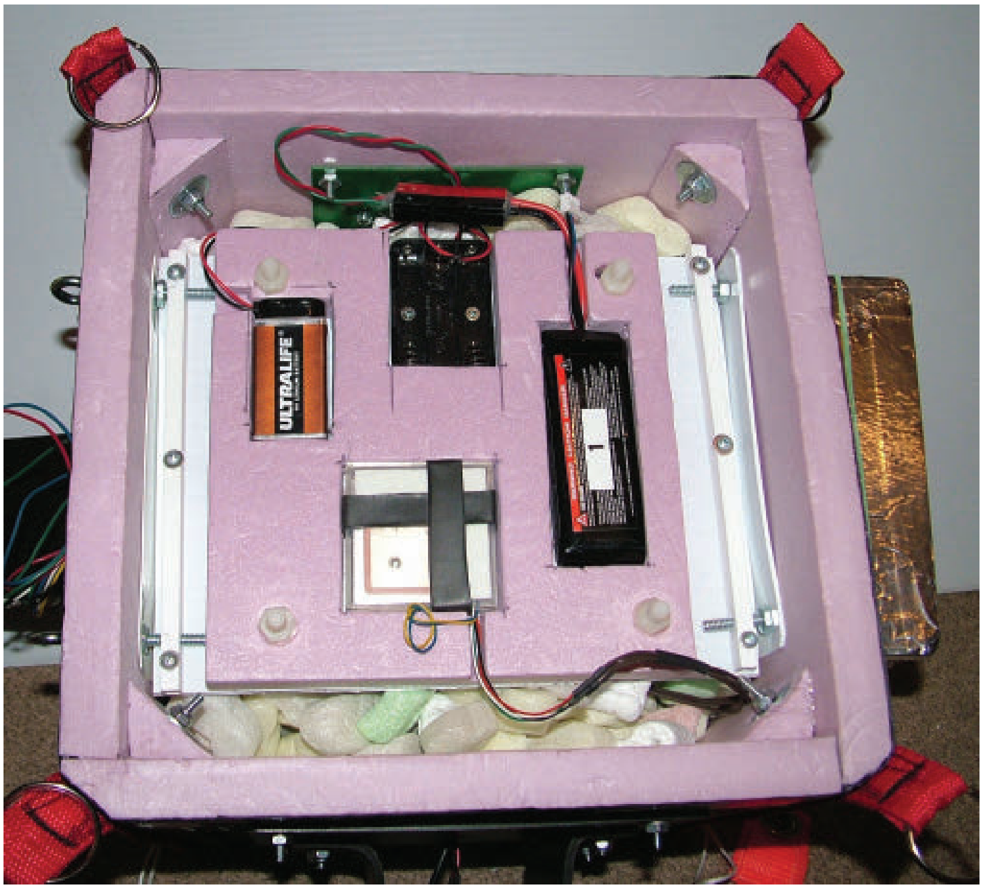
The final layer of the pallet is the lid which is another sheet of Coroplast. It is bolted over the top of the pallet, and covers the GPS and batteries. The nylon bolts holding the bottom of the pallet together are long enough to also bolt the lid down.

Attaching the GPS/ Battery Pallet to the Airframe

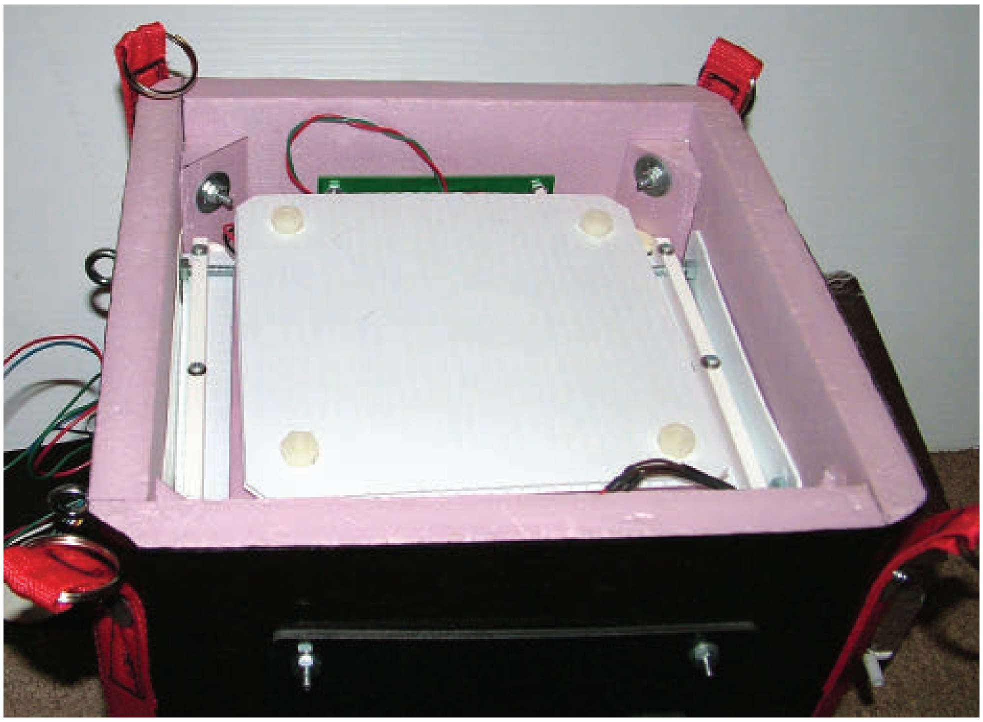
Prior to placing the GPS/battery pallet inside the airframe, I fill the volume between the avionics pallet and the GPS/battery pallet with Styrofoam packing peanuts. The peanuts transfer the weight of the GPS/battery pallet to the entire bottom of the airframe.

Distributing the pallet's weight to the airframe removes the need for the pallet to support the weight of the GPS and batteries during the peaceful ascent. However, it is more important during descent when chaos will repeatedly slam the pallet down.

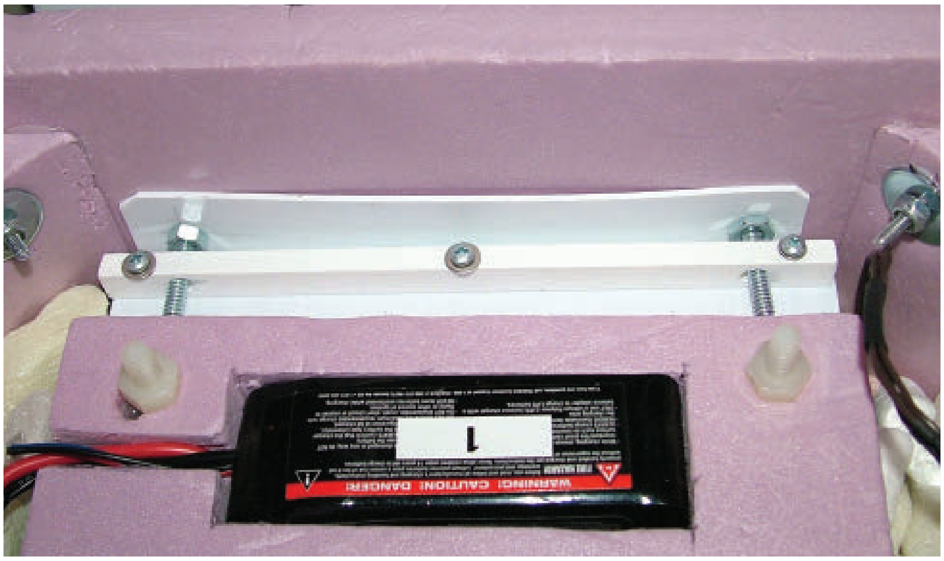
The GPS/battery pallet bolts



The battery/GPS pallet with its lid removed. Each item, the GPS, and assorted batteries are placed inside of a form-fit pocket.



Now, you can't see them. A second sheet of Coroplast is bolted over the batteries and GPS receiver. Four nylon bolts and nuts secure the lid into place over the pallet.



Two long 6-32 bolts capture the sides of the pallet to the airframe. There's another pair of bolts on the other side of the pallet that aren't shown in this image.

to opposite sides of the airframe. These bolts perform double-duty. First, they hold quad panels to the airframe, then second, they hold the pallet in place.

Quad panels are plastic plates that attach experiments to the sides of the airframe. Two of the quad panels on opposite sides of the airframe have extra long top bolts (there are four bolts per quad panel). These bolts are long enough to extend another inch inside the airframe and capture the GPS/battery pallet.

The bolts used to capture the GPS/battery pallet are actually eyebolts. The eye portion of the eyebolt is located on the outside of the airframe. So, to remove the GPS/battery pallet, I back out the eyebolt by unscrewing it while preventing the nut from spinning.

Since there are four eye bolts in the new airframe (two on each side), it is next to impossible for the GPS/battery pallet to come loose during a mission. So, why use eyebolts for securing the GPS/battery pallet? It's for the hatch.

Hatch

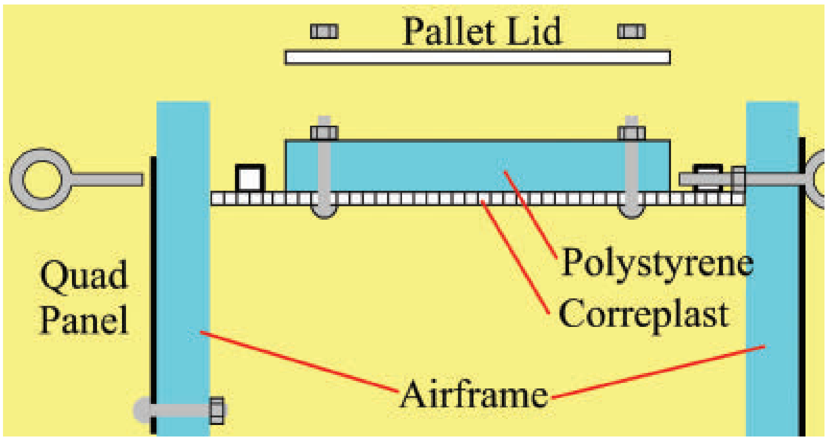
The last modification to my near space airframes is the hatch. The old hatch design used four straps of Velcro™ to secure it to the airframe. For over 80 missions, it was just those four straps that held everything inside the airframe.

Now, two 10 inch long bungee cords fasten the hatch to the airframe using the eyebolts that capture the GPS/battery pallet.

The first hook of the bungee cord connects to one eyebolt protruding from the side of the airframe. It then stretches over the hatch and over the top of the airframe.

The second hook of the bungee connects to an eyebolt located on the other side of the airframe. A second bungee cord does the same so that the hatch is held in place by the tension of the two cords.

As long as the eyebolts can't pull free, the cords hold the hatch down on the airframe more strongly than the Velcro straps. Plus, the bungee cords are fast connectors, so it only takes a few seconds to open and close the hatch.



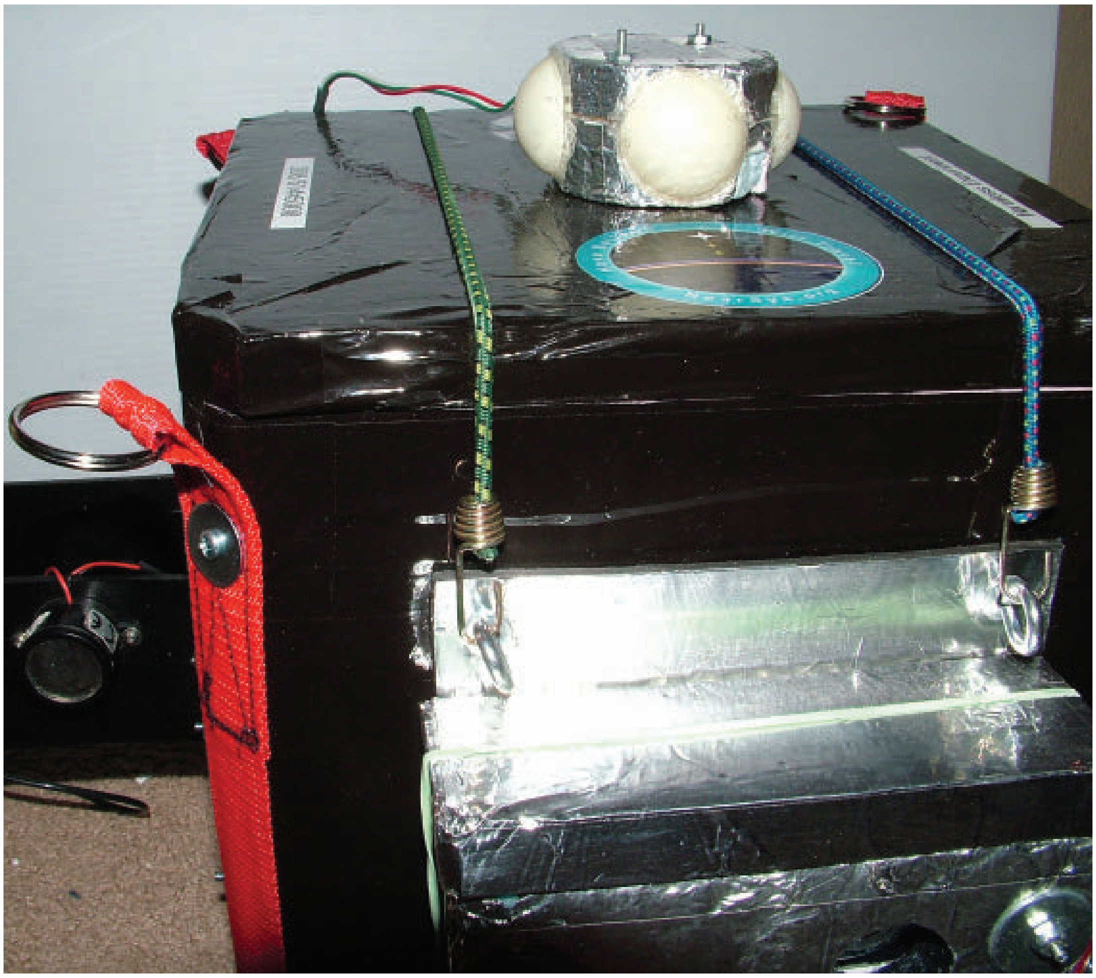
A cross-section of the GPS/battery pallet and how it's connected to the airframe.

Final Approach

That does it for the new airframe design. As I write this month's column, I'm experimenting with ion chambers as an affordable means to measure radiation in near space. I should be able to let you know the results of my experiments next time.

Also, don't forget about the Great Plains Super Launch. Zack Clobes of Project: Traveler is this year's host. The conference is open to anyone interested in learning more about amateur near space exploration.

So, if you're free June 12th and 13th, plan to visit us in Hutchinson, KS. I might just put you to work filling

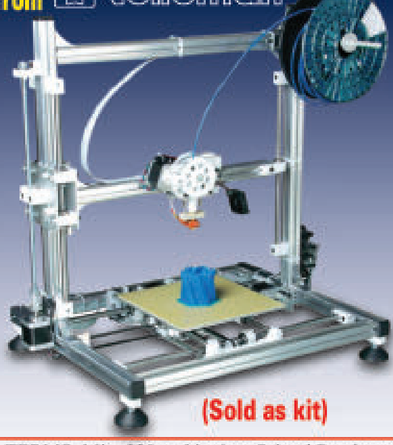


Two eyebolts that are holding the quad panel in place are also holding the hatch down.

a balloon. There's more information available at **www.superlaunch.org**.

Onwards and Upwards,
Your near space guide **NV**

NEW! from velleman



3D PRINTER KIT

Model K8200

Advanced kit requires mechanical, electronics and computer skills

Build-it-yourself 3D printer. Prints objects of maximum size of 20 x 20 x 20 cm using either PLA or ABS filament (3mm.) Extremely fast, reliable and precise even when printing at high speeds. Model K8200 is compatible with RepRap software and firmware (Free download). Sturdy aluminum construction, with heated print bed. Nine different colors of PLA filament available at $39.00. Complete with power supply and all necessary parts. Filament sold separately.

(Sold as kit)

NOW ONLY $795.00
Part No. 32VKK8200M

Electronix EXPRESS

TERMS: Min. $20 + shipping. School Purchase Orders, VISA/ MC, Money Order, Prepaid. NO PERSONAL CHECKS, NO COD. NJ Residents: Add 7% Sales Tax.

In NJ: 732-381-8020
FAX: 732-381-1006

900 Hart Street • Rahway, NJ 07065
800-972-2225

http://www.elexp.com
email: electron@elexp.com

ELECTRONET

CORIDIUM
ARM controllers running floating point BASIC from $10.00
www.coridium.us

GET THE NUTS AND VOLTS DISCOUNT!
Mention or enter coupon code NVRMZ142 and receive 10% off your order!
ramsey
www.ramseykits.com
AM/FM Broadcasters • Hobby Kits
Learnng Kits Test Equipment
...AND LOTS OF NEAT STUFF!

FREE PCB Design Software
Get Started Now at FabStream.com
FabStream
A DownStream Technologies Solution

THE ORIGINAL SINCE 1994
PCB-POOL®
Beta LAYOUT
• Low Cost PCB prototypes
• Free laser SMT stencil with all Proto orders
WWW.PCB-POOL.COM

HOBBY ENGINEERING
Kits, Parts and Supplies
www.HobbyEngineering.com

INVEST in your BOT!

INVEST in HiTEC
12115 Paine Street • Poway, CA 92064 • 858-748-6948 • www.hitecrcd.com

MyRO
www.myropcb.com
PCB, PCBA and More
LOW cost with HIGH Quality
1-888-PCB-MYRO

www.CHiPdesign.ca
PS/2 Keyboard to ASCII Converters
$16
Also with Ctrl-Alt-Del reset output!

ALL ELECTRONICS CORPORATION
Electronic Parts and Supplies.
www.allelectronics.com
Free 96 page catalog 1-800-826-5432

For the ElectroNet online, go to
www.nutsvolts.com
click Electro-Net

USB
Add USB to your next project--
It's easier than you might think!
DLP Design
USB-FIFO • USB-UART • USB/Microcontroller Boards
RFID Readers • Design/Manufacturing Services Available
Absolutely NO driver software development required!
www.dlpdesign.com

MOBILE APP NOW AVAILABLE!
SERVO
Download NOW On Your Favorite Mobile Device!
FOR ROBOT BUILDERS
iOS • ANDROID • KINDLE FIRE

superbrightleds.com
COMPONENT LEDs • LED BULBS • LED ACCENT LIGHTS

♦LED modules
♦Solar controls
♦Dusk to dawn
♦Dimmers12/24V
J2 LED LIGHTING LLC
www.j2ledlighting.com

NATIONAL RF, INC.
www.NationalRF.com
RF, DF, WxSat/Ham Gear, Dials
Attenuators, Plug-in Coils

■ BY FRED EADY

Setting up Linux with Bones is a Piece of Pi

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/march2014_DesignCycle.**

Over the years, I've muddled through various flavors of UNIX and AIX (IBM's version of UNIX). I would get hot on a UNIX project, learn what I needed, do the job, and forget about UNIX all together. DOS was the big hitter at the time for personal computing, and GUIs were usually not part of the OS. Back in those days, DOS applications provided the visuals. Everything else was either done via command line or batch file. Although UNIX was a green-screen OS, I considered DOS the lesser of the two evils. In my case, I guess that was because it was far easier to find something small that ran DOS than something small that ran UNIX.

These days, everyone is talking about how great Linux is. When I hear this talk, it takes me back to those old green-screen terminals and the operational terror I was exposed to. With the advent of portable Linux devices and their popularity, I decided to leave my fears in the past and give this Linux stuff a try.

Bones and Pis

The mainstream low cost mobile Linux platforms are known as BeagleBone Black and Raspberry Pi. Lots of folks have written lots of applications for both of them. However, I'm into doing things my way.

That means no matter how popular the BeagleBone Black and Raspberry Pi are, if I can't control them with my code on my bench, they are useless to me.

(From here on out, I will refer to these two platforms as Pi and Bone.)

I decided to see just how hard (or easy) it would be to set up a development environment based on the Pi and Bone. The goal was to have a single development environment that could be used for both devices. After all, they both can run a variant of Linux.

We used relatively expensive cross-compilers back in the old days, before C was an accepted way to program microcontrollers. I figured all of the open source stuff has got to be similar. Let's face it, in today's Linux mobile module environment, we're simply replacing that Z-80 and 8051 microcontroller with an ARM processor.

In that there are a multitude of ARM flavors available, cross-compilation has to be a part of the Pi and Bone development cycle.

Establishing a Base Camp

It would be a good idea to get some Bones and Pis, and load them with their respective Linux distributions first. Once we're sure the Linux kernels and associated drivers are really working, we can move on towards getting that development environment built.

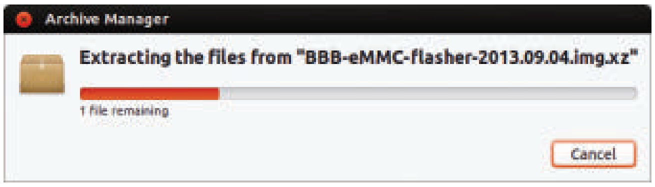
Both the Pi and Bone are capable of being directly attached to a standard PC monitor and keyboard. That's great if all we want to do is run someone else's code. We'll need a bit more access than that since we want to build our Linux applications in our personalized development environment and not on the actual target devices.

The idea is to use the superior processing power of a PC to quickly compile and download the application firmware to the target Bone or Pi. If you stand back and look at it, we did exactly that with those Z-80s and 8051s.

This is a Linux thing. So, I dug up an old ThinkPad whose battery charging circuitry had failed and replaced its native Windows XP Pro load with Ubuntu 13.04. I was pleasantly surprised. The Ubuntu desktop looked a lot like a Windows desktop.

Linux versions of Adobe Acrobat Reader and Firefox allowed me to read the Bone and Pi PDF documentation and access the Web for downloads and technical information. The "new" Linux computer also recognized all of the Windows computers on the EDTP workshop LAN and vice versa.

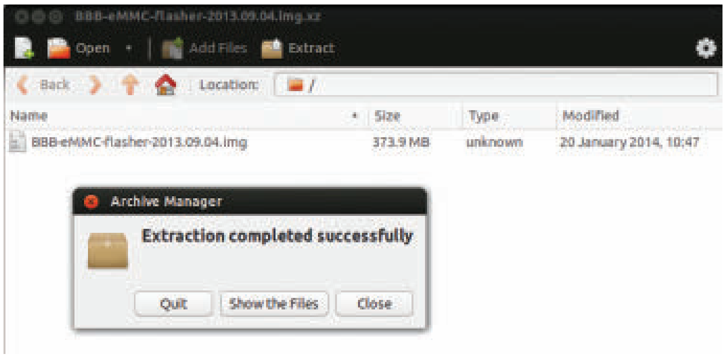
Despite all of the Windows "compatibility," I can still dive as deeply into the native Linux distribution assets as I need to. This isn't the snooty UNIX I remember. This is actually pretty cool!


Archive Manager

Extracting the files from "BBB-eMMC-flasher-2013.09.04.img.xz"

1 file remaining

■ Screenshot 1. Linux is "family oriented," no matter the size of the installation or the complexity of the hardware.

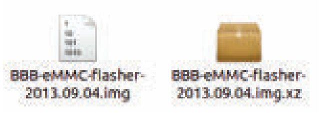

BBB-eMMC-flasher-2013.09.04.img.xz

Name	Size	Type	Modified
BBB-eMMC-flasher-2013.09.04.img	373.9 MB	unknown	20 January 2014, 10:47

Archive Manager

Extraction completed successfully

■ Screenshot 2. This looks familiar, does it not? It looks like Windows but smells like Ubuntu.


BBB-eMMC-flasher-2013.09.04.img
BBB-eMMC-flasher-2013.09.04.img.xz

■ Screenshot 3. Believe it or not, this is the Linux directory view. As time moved on, I found myself becoming very comfortable with Ubuntu and Linux to the point of performing some tasks with the Linux machine that I normally do with my Windows laptop.

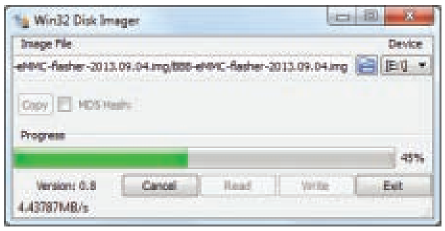

Win32 Disk Imager

Image File: -eMMC-flasher-2013.09.04.img/BBB-eMMC-flasher-2013.09.04.img

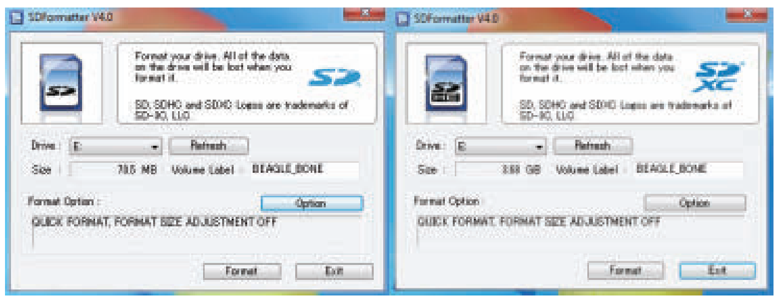
Device: [E:\]

Progress: 45%

Version: 0.8

4.43787MB/s

■ Screenshot 5. This tool is specifically designed to read and write raw disk images.


SDFormatter V4.0

Format your drive. All of the data on the drive will be lost when you format it.

SD, SDHC and SDXC Logos are trademarks of SD-3C, LLC.

Drive: E:

Size: 785 MB Volume Label: BEAGLE_BONE

Format Option:

QUICK FORMAT, FORMAT SIZE ADJUSTMENT OFF

SDFormatter V4.0

Format your drive. All of the data on the drive will be lost when you format it.

SD, SDHC and SDXC Logos are trademarks of SD-3C, LLC.

Drive: E:

Size: 3.68 GB Volume Label: BEAGLE_BONE

Format Option:

QUICK FORMAT, FORMAT SIZE ADJUSTMENT OFF

■ Screenshot 4. Moving from left to right, you can see that the SDFormatter Windows application got everything it could out of the 4 GB microSD card.

Loading the BeagleBone Black

The Bone comes loaded with the Angstrom distribution. So, instead of experimenting with a different Linux distribution, we will walk down the well-known Angstrom road. That doesn't mean we can't repave it before we set out. I proceeded to download the latest Angstrom distribution. Let's walk through the steps I took to install it.

Unlike the Pi, the Bone has the ability to store the Linux kernel in onboard non-volatile Flash. With that, the Bone images I downloaded must be processed to make them suitable for installation. The Bone Angstrom distribution downloads as *Bone-eMMC-flasher-2013.09.04.img.xz*. We can use Linux or Windows to perform the image conversion required to load the Bone. Let's use Linux this time around.

We can convert the downloaded image using the Ubuntu Archive Manager. Linux prompts me by asking if I want to use Archive Manager, and I agree. This kicks off the process captured in **Screenshot 1**. As you can see in **Screenshot 2**, the result is a 373.9 MB Flashable image called *Bone-eMMC-flasher-2013.09.04.img*.

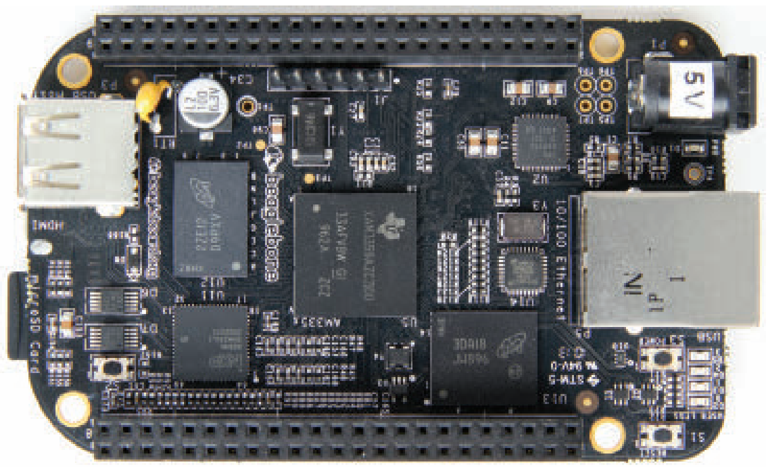
The Bone is equipped with a microSD interface that is primarily intended for loading Flashable Linux images. With a little bit of extra work, the microSD card can also be used to store data. We'll get to that after we do what we've set out to do here.

Our newly converted image must be moved from our Linux-based PC to a microSD card. Right now — as **Screenshot 3** depicts — the *Bone-eMMC-flasher-2013.09.04.img* file resides on the Linux PC. It is possible to move the file to a microSD card using Linux, but it is a much simpler process to use a couple of Windows applications. Recall that the Linux machine speaks fluently with Windows on the EDTP LAN, so I used LAN resources to copy the *Bone-eMMC-flasher-2013.09.04.img* file over to my Windows laptop.

Before we can move the Angstrom image to the microSD card, we must be sure that the card is properly formatted. Some of the best computing things in life are free, as is the SDFormatter application featured in **Screenshot 4**. I unleashed SDFormatter on a 4 GB microSD card. I started with the size value shown in the left window and ended up with the size value you see in the right window.

Once the microSD card has been properly formatted, we can place the Bone Angstrom image on the microSD

■ Photo 1. This is an aerial view of my BeagleBone Black. What you don't see are the microSD, HDMI, and USB sockets which are mounted on the other side of the printed circuit board.



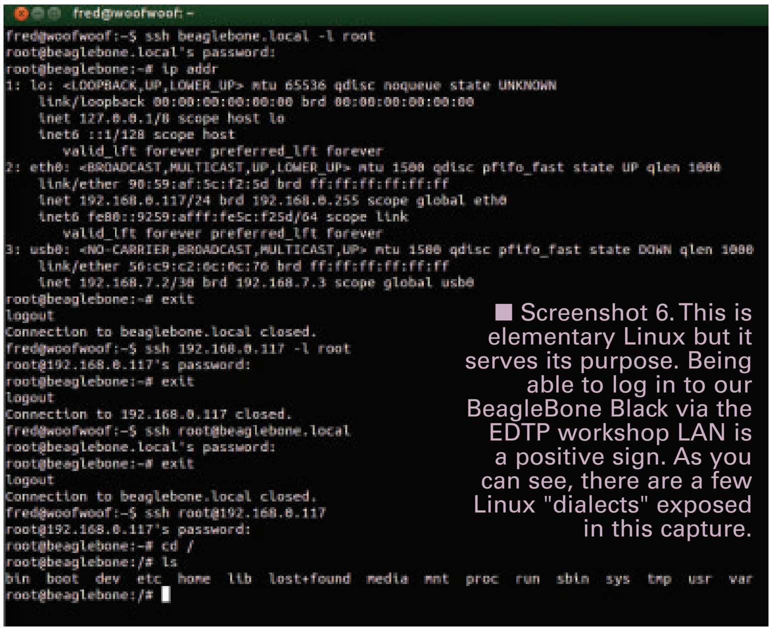
card. This involves a bit more than just a copy operation. We must rely on the services of yet another free utility called Win32 Disk Imager. **Screenshot 5** shows Win32 Disk Imager at work, moving the raw disk image from the Windows laptop to the 4 GB microSD card.

Now that we have a good Bone disk image on the microSD card, let's install Angstrom. All we have to do is remove power from the Bone, insert the microSD card containing our Angstrom image, hold down the boot switch, and apply power.

After a few seconds, the Bone's bank of LEDs will begin to flash. At that point, you can release the Bone boot switch and the Flashing process will continue. All four of the Bone's LEDs will illuminate solidly when the Flashing has completed. You can see the boot switch in the lower left corner of **Photo 1** and the LED bank in the rightmost bottom corner of **Photo 1**.

There are some basic networking things that Angstrom and the Bone just do. I'll attach the newly loaded Bone to the EDTP workshop LAN. I should be able to log on to the Bone from the Linux laptop, which is also on the EDTP workshop LAN. Check out **Screenshot 6**. I've opened up a terminal window on the Linux laptop and used Secure Shell (SSH) to remotely log on to the Bone. I issued *ip addr* to obtain the Bone's IP address. The Bone returned its loopback address (127.0.0.0) and the eth0 interface IP address (192.168.0.117).

Now that we know the Bone's IP address, the *beaglebone.local* can optionally be replaced with the Bone's IP address at login time. As you have already ascertained, using the *beaglebone.local* method is the easier login route.

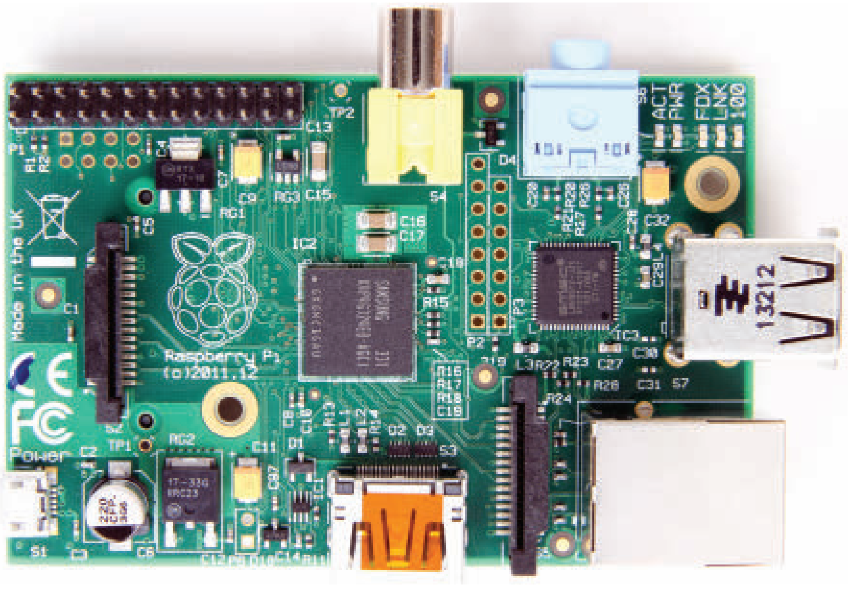


```
fred@woofwoof: ~
fred@woofwoof:~$ ssh beaglebone.local -l root
root@beaglebone.local's password:
root@beaglebone:~# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 90:59:af:5c:f2:5d brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.117/24 brd 192.168.0.255 scope global eth0
    inet6 fe80::9259:afff:fe5c:f25d/64 scope link
       valid_lft forever preferred_lft forever
3: usb0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 56:c9:c2:0c:0c:70 brd ff:ff:ff:ff:ff:ff
    inet 192.168.7.2/30 brd 192.168.7.3 scope global usb0
root@beaglebone:~# exit
logout
Connection to beaglebone.local closed.
fred@woofwoof:~$ ssh 192.168.0.117 -l root
root@192.168.0.117's password:
root@beaglebone:~# exit
logout
Connection to 192.168.0.117 closed.
fred@woofwoof:~$ ssh root@beaglebone.local
root@beaglebone.local's password:
root@beaglebone:~# exit
logout
Connection to beaglebone.local closed.
fred@woofwoof:~$ ssh root@192.168.0.117
root@192.168.0.117's password:
root@beaglebone:~# cd /
root@beaglebone:/# ls
bin  boot  dev  etc  home  lib  lost+found  media  mnt  proc  run  sbin  sys  tmp  usr  var
root@beaglebone:/# 
```

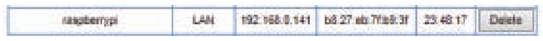
■ Screenshot 6. This is elementary Linux but it serves its purpose. Being able to log in to our BeagleBone Black via the EDTP workshop LAN is a positive sign. As you can see, there are a few Linux "dialects" exposed in this capture.

Loading the Raspberry Pi

Let's turn our attention to the Pi. I happen to have the 512 MB version

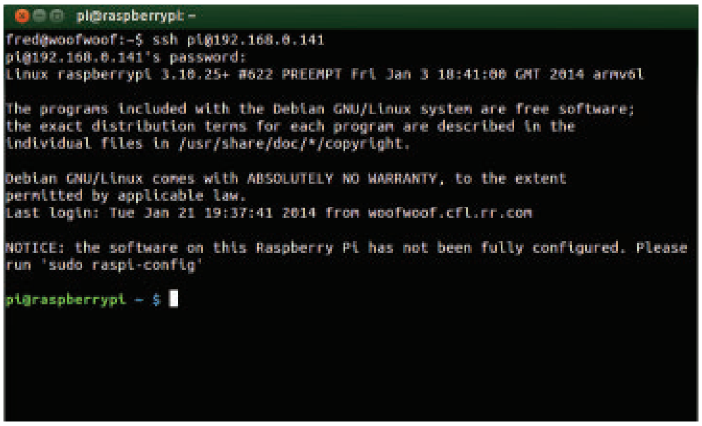


■ Photo 2. The Raspberry Pi and BeagleBone Black are first cousins. It's up to you to decide which is the ugly one. Beauty is truly in the eye of the beholder. Or should I say, beauty depends on the beholder's application.



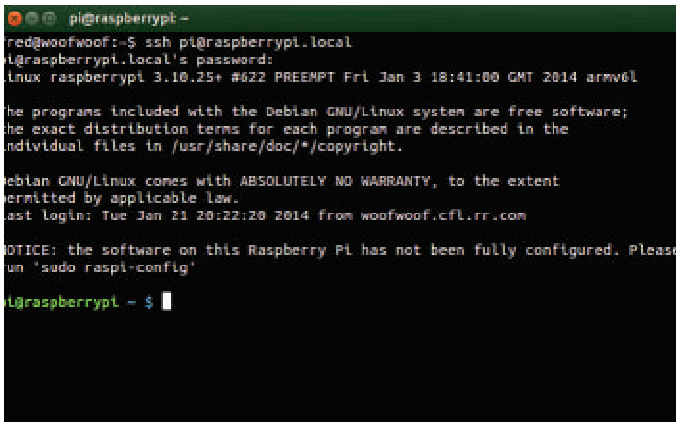
raspberrypi	LAN	192.168.0.141	b8:27:eb:7f:b8:3f	23:48:17	Delete

■ Screenshot 7. This is a singular capture of the EDTP workshop router's DHCP table. The EDTP workshop router has assigned 192.168.0.141 to our Raspberry Pi.



```
fred@woofwoof:~$ ssh pi@192.168.0.141
pi@192.168.0.141's password:
Linux raspberrypi 3.10.25+ #622 PREEMPT Fri Jan 3 18:41:00 GMT 2014 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jan 21 19:37:41 2014 from woofwoof.cfl.rr.com

NOTICE: the software on this Raspberry Pi has not been fully configured. Please
run 'sudo raspi-config'

pi@raspberrypi ~ $
```

■ Screenshot 8. We're up with the Raspberry Pi! Before we can go any further, we'll finish up the suggested configuration operation and do some updating.



```
red@woofwoof:~$ ssh pi@raspberrypi.local
i@raspberrypi.local's password:
inux raspberrypi 3.10.25+ #622 PREEMPT Fri Jan 3 18:41:00 GMT 2014 armv6l

he programs included with the Debian GNU/Linux system are free software;
he exact distribution terms for each program are described in the
ndividual files in /usr/share/doc/*/copyright.

ebian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
ermitted by applicable law.
ast login: Tue Jan 21 20:22:20 2014 from woofwoof.cfl.rr.com

OTICE: the software on this Raspberry Pi has not been fully configured. Please
un 'sudo raspi-config'

i@raspberrypi ~ $
```

■ Screenshot 9. Pretty cool stuff, this Linux. By issuing a simple command that ran an Internet errand for us automagically, we can now remotely log in to our Raspberry Pi in the same way we remotely log in to our BeagleBone Black.

(Model B) on the bench. As you can see in **Photo 2**, the Pi is radically different as far as physical design is concerned. There are some logical differences too. The Pi boots from a microSD card only. No big deal because loading the latest version of Raspbian onto the Pi's microSD card is procedurally similar to that of the Bone. SDFormatter is used to "clean" an 8 MB microSD card and Win32 Disk Imager puts down the Raspbian image. The microSD card is then physically mounted on the Pi and power is applied.

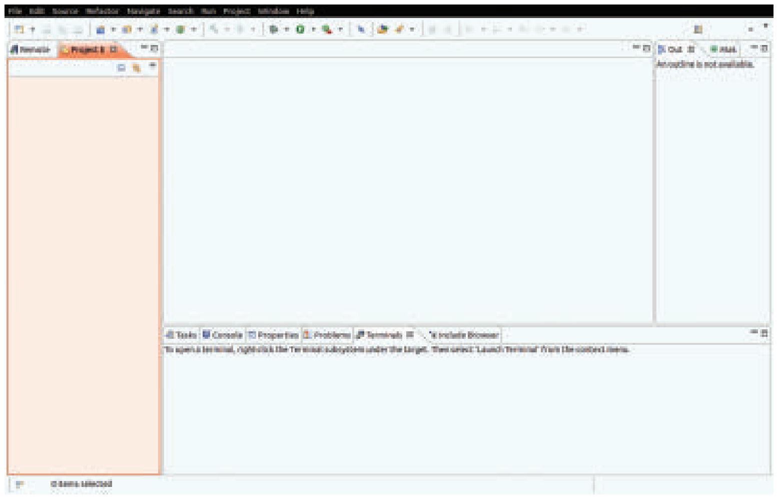
Getting at the Pi the first time around without a monitor and keyboard is not very straightforward. We must first determine the Pi's IP address. Since the Pi defaults to DHCP operation, the easy way to obtain the Pi's IP address is to check the router's DHCP table.

Our Pi was assigned 192.168.0.141 as evidenced in **Screenshot 7**. Recall that we were able to get at our Bone without specifying the IP address using *beaglebone.local*. Well, guess what? Linux is Linux is Linux. We can do the same with the Pi. Before I put *raspberrypi.local* into place, I'll perform the operation that has been "suggested" in **Screenshot 8**. For now, I will only select the *Expand Filesystem* option to resize the root partition.

In my Linux travels, I learned that it is good to bring a new installation up to snuff using the update and upgrade methods. For the Pi, that equates to *sudo apt-get update* and *sudo apt-get upgrade*, respectively. Once that was done, I issued *sudo apt-get install avahi-daemon*.

In a nutshell, Avahi is a system based on the mDNS/DNS-SD protocol suite that enables service discovery on a LAN. This allows us to simply attach the Pi to our LAN and find it using the *raspberrypi.local* domain.

Screenshot 9 verifies that the Pi is ready to go to work. By the way, *sudo* or "su do" is a program that allows users to issue certain root (super user) commands without having full root privileges.



File Edit Source Refactor Navigate Search Run Project Window Help
An outline is not available.
Tasks Console Properties Problems Terminals
To open a terminal, right click the Terminal subsystem under the target. Then select 'Launch Terminal' from the context menu.
0 items selected

■ Screenshot 10. Installing Eclipse and RSE was painless as the Eclipse IDE and RSE installation procedures are very well documented in a number of locations on the Internet.

Living On the LAN

Our Bone and Pi need a common launch pad. In our case, that launch pad — or Integrated

The Raspberry and BeagleBone can be had from a number of distributors. You can source them from your preferences of distributors. Here's a few I know of:

Raspberry Pi:
Newark
AdaFruit
Newegg
Allied
MCM

Beaglebone Black:
Newark
Digi-Key
Mouser

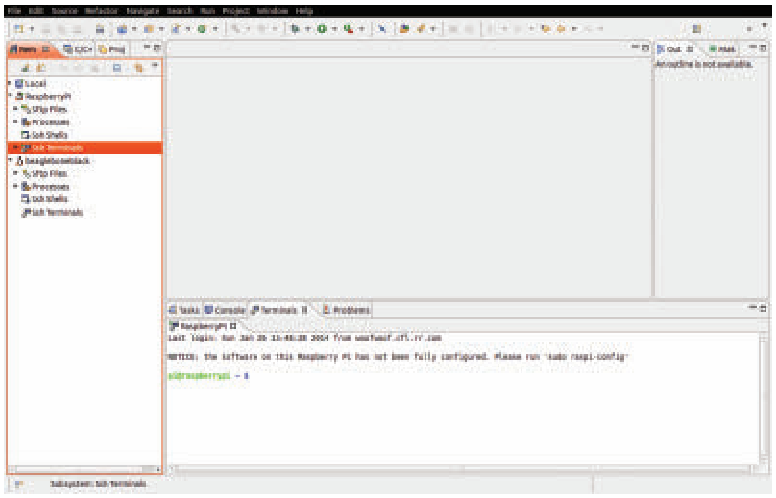
Development Environment — is called Eclipse. Eclipse is an open source IDE that runs under Windows and Linux. To install Eclipse on the Linux laptop, I opened a terminal window and issued *apt-get install eclipse*. Once Eclipse installed, the next step was to install the Eclipse C++ Development Tools.

This is done by clicking on Help and choosing the Install New Software in the dropdown menu. I then chose Work with: Indigo Update Site in the Install window. All of the available software was listed, and I selected C++ Development Tools and C/C++ Library API Documentation Hover Help from the list. Following the install process, the Eclipse IDE was primed to do some C/C++ development. Since I wanted to work with the Pi and Bone at the end of a wire, I also installed the Remote System Explorer plugin. RSE allows us to look at the Pi and Bone file systems, transfer files between hosts on the LAN, and generally command and control hosts that we can connect to.

Once everything was installed, I was left with the Eclipse IDE interface you see in **Screenshot 10**. Note that the Eclipse IDE interface includes a Project Explorer tab and a Remote tab.

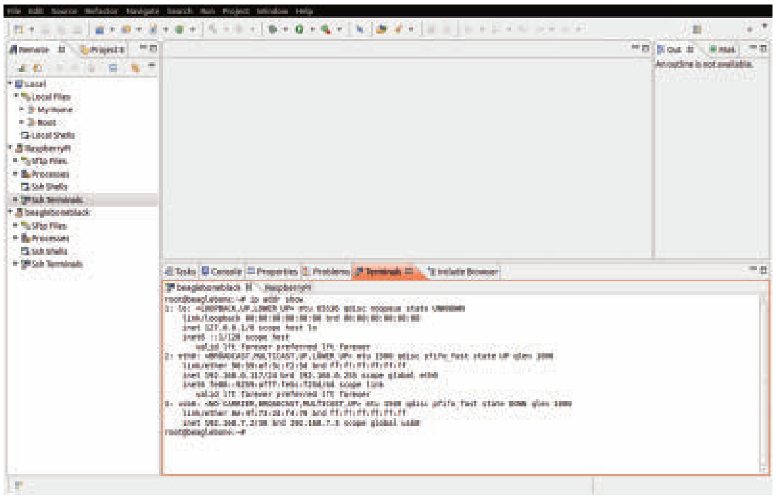
Let's see if we can contact and connect the Pi to the Eclipse IDE. A right click anywhere inside the Eclipse Remote tab area brings up the New Connection menu item. We will use simple logic to make our choices as we work our way through the various configuration windows. We know that we are working with a Linux system. We also know that we have established *.local* domains for the Pi and Bone. So, we will address the Pi as *raspberrypi.local* and call the new connection RaspberryPI. We are basically interested in communicating with the Pi via SSH access. With that, we will focus on configuring only the SSH terminals and connections.

In **Screenshot 11**, you can see that I have successfully configured an Eclipse remote connection to the Pi called *RaspberryPI*. I also launched an SSH terminal session that logged in as *pi*. I applied the same procedures I used to form the Pi connection to the Bone. I was able to launch a *root* SSH terminal session for the Bone, which is coexisting with the Pi *pi* terminal session in **Screenshot 12**.



```
Last login: Sun Jan 26 13:48:28 2014 from [illegible]
NOTICE: the software on this Raspberry Pi has not been fully configured. Please run 'sudo raspi-config'
pi@raspberrypi ~ $
```

■ Screenshot 11. Success! The Raspberry Pi is connected to the Eclipse IDE, and now falls under the control of the C/C++ development environment we have established. The terminal was kicked off with the SSH Terminal's Launch Terminal menu selection.



```
root@beaglebone:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether [illegible] brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.117/24 brd 192.168.0.255 scope global eth0
    inet6 [illegible]/64 scope link
       valid_lft forever preferred_lft forever
3: usb0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether [illegible] brd ff:ff:ff:ff:ff:ff
    inet 192.168.7.2/30 brd 192.168.7.3 scope global usb0
root@beaglebone:~#
```

■ Screenshot 12. The same process used to connect to the Raspberry Pi was applied for the BeagleBone Black. Take note of the Pi terminal session that is coexisting with the new BeagleBone terminal session. We can now move files between the Bone, Pi, and Linux PC.

Next Time

We are ready to start writing Bone and Pi applications using our Eclipse-based cross-compiler environment. So, be sure to get your Linux host computer in touch with your Bone and/or Pi before we talk again. In the next installment of Design Cycle, we are going to jump into Pi and Bone application development. **NV**

Continued from page 7

Data Module at 315-434 MHz.'

3. Locate the product box at the bottom of the web page. Don't get scared when you see the cost of shipping listed below the product box. The 'Place Order' page has a drop-down menu for selecting a lower shipping cost.

4. Click on 'Place Order - Shopping Cart.'

5. Click on the link 'RF Products' at the top of the page.

6. Select your product and then click 'Check Out.'

OPTION #2
(Note: Minimum $40 order required)
http://laipac.com
Voice USA: 1-800-897-0525
Voice Canada: 905-762-1228
TLP434A Transmitter = $4.80
RLP434 Receiver = $4.80

Just click on the 'GPS & RF Components' tab on their home page and select Transmitter/Receiver modules.

OPTION #3
(NOTE: Minimum $50 order required)
www.abra-electronics.com
ABRA Electronics, Inc.
1320 State Route 9, #21
Champlain, NY 12919
Tel: 1-800-717-ABRA(2272)
Fax: 1-800-898-ABRA(2272)

Just type WRL-10534 into the search box for the 434 MHz transmitter at $3.95 each, or type in WRL-10532 for the 434 MHz receiver at $4.95 each.

OPTION #4
www.sparkfun.com
Front Desk: 303-284-0979
(Monday through Friday, 9 am to 5 pm, Mountain Time)
WRL-10532 434 MHz Receiver = $4.95 each
WRL-10534 434 MHz Transmitter = $3.95 each

1. Click the Wireless link on the side menu.

2. Scroll down and select the transmitter and receiver.

The antennas for the alarm system can be purchase from the following sources:

www.amazon.com
$3.50 each
Type "430-470 MHz UHF two-way radio whip" into the search box on the home page.

www.ebay.com
$3.55 each
Type "UHF Antenna NAE6483AR for MOTOROLA GP340" into the search box on the home page.

Planting a Seeed

Regarding Bryan Bergeron's editorial entitled, "Sound Off!" in the November 2013 issue — he may or may not be aware of a more inexpensive MP3 module from SeeedStudio. It has a lot of features for $9.90. You can find out more about it at **www.seeedstudio.com/depot/garan-audio-module-p-1607.html**.

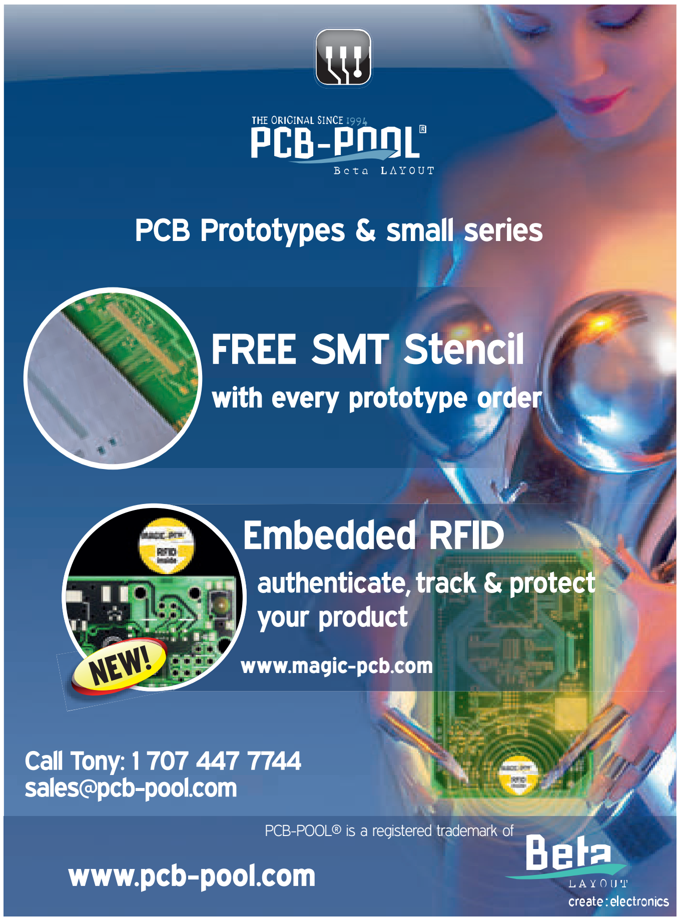
Alan Vogel

I'm a big fan of SeeedStudio gear — especially the grove system. Thanks for reminding me.

Bryan Bergeron
Editor

Great Bits of Info

Thank you for the "All About Clocks for PICs" article and examples in the January 2014 issue. Readers who enjoy PIC bit-banging might enjoy my PIC PWM manual, posted on the Sierra Radio Systems site under Project Gallery at **www.hamstack.com/pwm.html**. Code examples use the free version of Swordfish Basic. These 40 plus examples use the PIC18F4620 and PIC18F46K22 chips, and the HamStack boards from Sierra Radio Systems. However, it is easy to breadboard a "bare-bones"


THE ORIGINAL SINCE 1994
PCB-POOL®
Beta LAYOUT
PCB Prototypes & small series
FREE SMT Stencil
with every prototype order
Embedded RFID
authenticate, track & protect your product
NEW!
www.magic-pcb.com
Call Tony: 1 707 447 7744
sales@pcb-pool.com
PCB-POOL® is a registered trademark of
Beta LAYOUT
create:electronics
www.pcb-pool.com

system as shown in another manual in the Project Gallery – *PIC18F4620 Cookbook, Hands-on Experiments*, page 69: "Minimum Circuit Schematic: PICkit™ 2 programmer & 18F4620." Many of the bit-banging examples would work with other chips in the 18F series.

Thomas Henry hit that nail on the head when he said "Datasheets for PICs run in the hundreds of pages and aren't necessarily organized for best learning." I wrote these notes as I constantly turned back and forth through the 500 plus pages of several Microchip datasheets, trying to get any PWM to show up on my scope. My notes include numerous page references to Microchip datasheets to help the DIYer get into these complex documents. A YouTube video exhibits some of these experiments at **www.youtube.com/watch?v=bDlDZJNax5I**.

As a long time reader and subscriber to *Nuts & Volts*, I included a tribute to "Smiley" Joe Pardue in the Cookbook page 77, using the WDT to cycle cylon eyes.

Robert Ralston

Thanks for the note and the information for N&V readers. Obviously a labor of love.

Bryan Bergeron

Constructive Construction

I am writing to say that I read the article by Robert Reed (Construction of a Low Budget 180 MHz RF Sweep Generator) in the December 2013 issue with interest because I have been trying to get more and better output from my Heathkit IG5280 with no success to-date. I plan to build this project using surface-mount parts and wind my own coils.

I want to comment on the article by Robert Atkinson (Improved Efficiency 13.8V Power Supply) in the February 2014 issue. I applaud the way he gives the rational for selection of components and tips on construction. I think the author should have mentioned that the MOSFET turn-on voltage Vgs(on) should be matched. Otherwise, one may hog all the current and defeat the purpose of paralleling. If all the MOSFETs are from the same batch, they likely will be matched.

Russell Kincaid

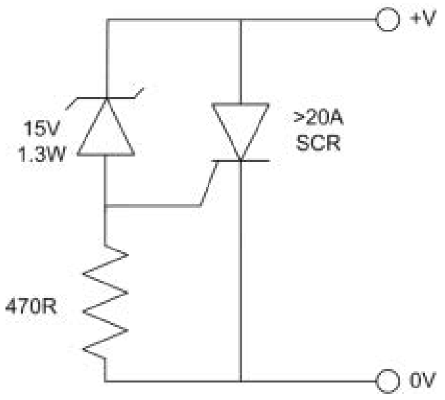
Use a Crowbar

The 13.8V MOSFET Power Supply article in the February 2014 issue was great! There is something missing from the design, however. If there is a failure that results in an overvoltage — such as a shorted FET — then the raw DC will appear at the output. This could be a disaster to a high value load like my ham transceiver. A crowbar circuit should be added to quench the supply and blow the fuse so the load is protected.

Dave Hartman

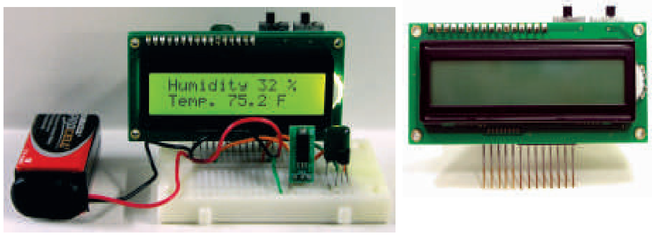
Thank you for your comments. A crowbar circuit was considered, but I felt it wasn't necessary. This is because of the very conservative rating of the MOSFETs and the low unregulated voltage. A bipolar design typically has a 19V to 21V transformer giving a unregulated voltage of 26V to 29V. As you say, a failure of the (often under-rated) pass transistors could be very damaging. The 15V transformer in the MOSFET design would give an unregulated output of 20V — much less likely to cause damage. However, if you wish to add a crowbar, a conventional circuit using a zener and SCR connected across the output will work just fine. Take a look at the circuit shown here.

Robert Atkinson G8RPI



Images Scientific Instruments Inc.

PIC Basic Project Board



16F88 PIC Basic Project Board Features
* LCD Display - back light & contrast control
* 2 A/D channels, 8 digital I/O lines, 5V and 9V operation
* Free student version of PICBASIC PRO & Microcode Studio with purchase of PIC Programming book bundle.
* 16F88 Programming Service Available
* Project library with source codes and hex files available.
* See 16F88 PCB Project page for more details

Sensors:	Do More:
Humidity	Frequency Meter
Temperature	Pulse Generator
Toxic Gas	Elapse Timer
Bend and Force Sensors	Radiation Pulse Counter
CdS Light sensors	Volt Meter

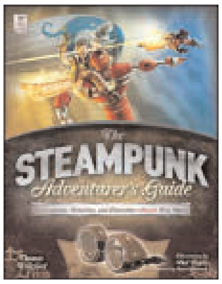
www.imagesco.com/microcontroller/16f88.html

For complete product details, visit our webstore!!

The Nuts & Volts WEBSTORE

GREAT FOR DIYers!

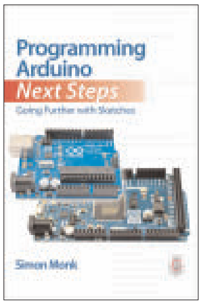
The Steampunk Adventurer's Guide
by Thomas Willeford



Steampunk stalwart Thomas Willeford cordially invites you on an adventure — one in which you get to build ingenious devices of your own! Lavishly illustrated by award-winning cartoonist Phil Foglio, *The Steampunk Adventurer's Guide: Contraptions, Creations, and Curiosities Anyone Can Make* presents 10 intriguing projects ideal for makers of all ages and skill levels, woven into an epic tale of mystery and pursuit.

$25.00

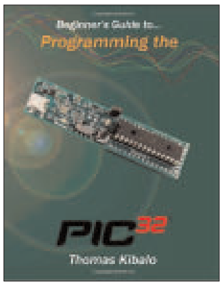
Programming Arduino Next Steps: Going Further with Sketches
by Simon Monk



Arduino guru Simon Monk reveals advanced programming techniques for Arduino! *Programming Arduino Next Steps: Going Further with Sketches* is the must-have follow-up to Monk's bestseller, *Programming Arduino: Getting Started with Sketches*. Aimed at experienced programmers and hobbyists who have mastered the basics, this book takes you "under the hood" of the Arduino, revealing professional-level programming secrets.

$20.00

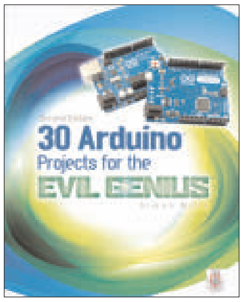
Beginner's Guide to Programming the PIC32
by Thomas Kibalo



Thomas Kibalo, who has written many articles for *Nuts & Volts Magazine* delivers the beginner's book many have been looking for: *Beginner's Guide to Programming the PIC32*. Using the low cost Microchip Microstick II module with built-in programmer, the socketed PIC32MX250F128B microcontroller, and the free to download version ofthe MPLAB XC32 compiler, Kibalo takes you step by step through the fundamentals of programming the PIC32.

Reg Price $39.95 **Sale Price $31.95**

30 Arduino Projects for the Evil Genius: Second Edition
by Simon Monk

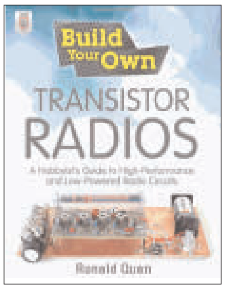


Fully updated throughout, this do-it-yourself guide shows you how to program and build fascinating projects with the Arduino Uno and Leonardo boards, and the Arduino 1.0 development environment. *30 Arduino Projects for the Evil Genius, Second Edition*, gets you started right away with the simplified C programming you need to know, and demonstrates how to take advantage of the latest Arduino capabilities.

$25.00

Build Your Own Transistor Radios
by Ronald Quan

A Hobbyist's Guide to High Performance and Low-Powered Radio Circuits

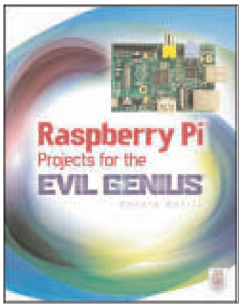


Create sophisticated transistor radios that are inexpensive yet highly efficient. Inside this book, it offers complete projects with detailed schematics and insights on how the radios were designed. Learn how to choose components, construct the different types of radios, and troubleshoot your work.

***Paperback, 496 pages**

$49.95

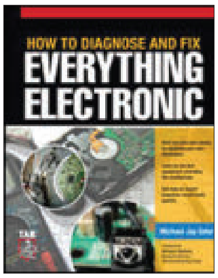
Raspberry Pi Projects for the Evil Genius
by Donald Norris



This wickedly inventive guide shows you how to create all kinds of entertaining and practical projects with the Raspberry Pi operating system and programming environment. Each fun, inexpensive Evil Genius project includes a detailed list of materials, sources for parts, schematics, and lots of clear, well-illustrated instructions for easy assembly. The larger workbook-style layout makes following the step-by-step instructions a breeze.

$25.00

How to Diagnose and Fix Everything Electronic
by Michael Jay Geier

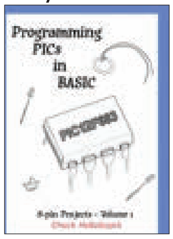


Master the Art of Electronics Repair

In this hands-on guide, a lifelong electronics repair guru shares his tested techniques and invaluable insights. *How to Diagnose and Fix Everything Electronic* shows you how to repair and extend the life of all kinds of solid-state devices, from modern digital gadgetry to cherished analog products of yesteryear.

$24.95

Programming PICs in Basic
by Chuck Hellebuyck



If you wanted to learn how to program microcontrollers, then you've found the right book! Microchip PIC microcontrollers are being designed into electronics throughout the world and none is more popular than the eight-pin version. Now the home hobbyist can create projects with these little microcontrollers using a low cost development tool called the CHIPAXE system and the Basic software language.Chuck Hellebuyck introduces how to use this development setup to build useful projects with an eight-pin PIC12F683 microcontroller. **$14.95**

Master and Command C for PIC MCUs
by Fred Eady

Master and Command C for PIC MCU, Volume 1 aims to help readers get the most out of the Custom Computer Services C compiler for PIC microcontrollers. The author describes some basic compiler operations that will help programmers particularly those new to the craft create solid code that lends itself to easy debugging and testing. As Eady notes in his preface, a single built-in CCS compiler call (output_bit) can serve as a basic aid to let programmers know about the "health" of their PIC code.

$14.95

Order online @ **www.store.nutsvolts.com**
Or CALL **1-800-783-4624** today!

BOOK & KIT COMBOS

Raspberry Three Book Combo

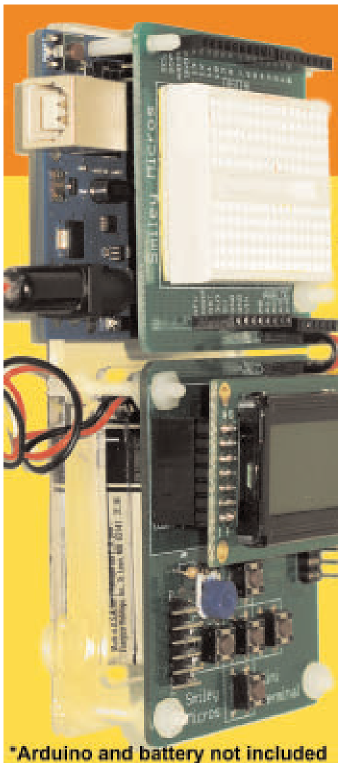
Programming the Raspberry Pi
Getting Started with Python
Simon Monk
Raspberry Pi User Guide
Raspberry Pi Projects for the EVIL GENIUS

Only $48.95
Plus
FREE Priority Mail Shipping
US Only
Offer Ends 2/28/14

Beginner's Guide Complete Combo!

Beginner's Guide to Embedded C Programming
VOL 3 Experiment Component Pack

Combo Price
$229.95
For complete details, visit our webstore @ www.nutsvolts.com.



Arduino Handheld Prototyper System

Arduino Proto Shield
$16.99

I2C Mini Terminal
$29.99

Handheld Base
$13.99

All three*: $59.99

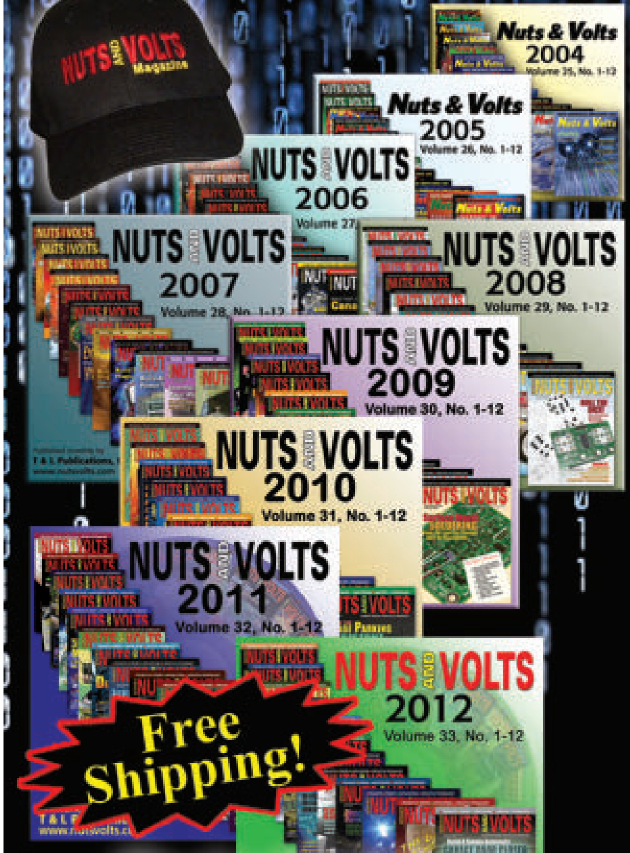
*Arduino and battery not included

From Smiley's Workshop

CD-ROM SPECIAL

Nuts & Volts 9 CD-ROMs & Hat Special!

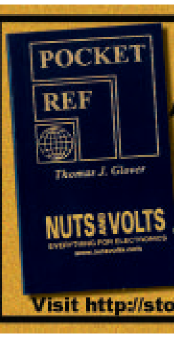
That's 108 issues.
Complete with supporting code and media files.



NUTS AND VOLTS Magazine
Nuts & Volts 2004 Volume 25, No. 1-12
Nuts & Volts 2005 Volume 26, No. 1-12
NUTS AND VOLTS 2006 Volume 27
NUTS AND VOLTS 2007 Volume 28, No. 1-12
NUTS AND VOLTS 2008 Volume 29, No. 1-12
NUTS AND VOLTS 2009 Volume 30, No. 1-12
NUTS AND VOLTS 2010 Volume 31, No. 1-12
NUTS AND VOLTS 2011 Volume 32, No. 1-12
NUTS AND VOLTS 2012 Volume 33, No. 1-12

Free Shipping!

Only $189.95
or $24.95 each.



The Nuts & Volts Pocket Ref
All the info you need at your fingertips!
This great little book is a concise all-purpose reference featuring hundreds of tables, maps, formulas, constants & conversions.
AND it still fits in your shirt pocket!

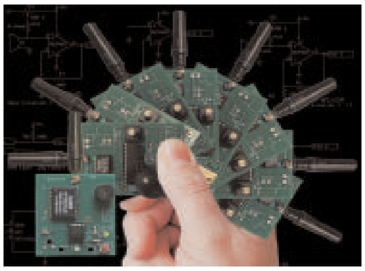
Only $12.95

Visit http://store.nutsvolts.com or call (800) 783-4624

CALL 1-800-783-4624 today!
Order online @ www.nutsvolts.com

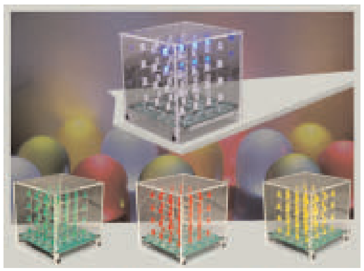
PROJECTS

Super Detector Circuit Set



Pick a circuit!
With one PCB you have the option of detecting wirelessly: temperature, vibration, light, sound, motion, normally open switch, normally closed switch, any varying resistor input, voltage input, mA input, and tilt, just to name a few.
Subscriber's Price **$32.95**
Non-Subscriber's Price **$35.95**

3D LED Cube Kit

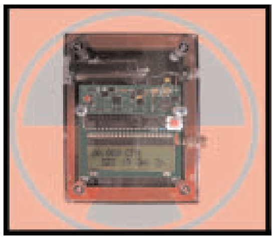


This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue
Subscriber's Price **$57.95**
Non-Subscriber's Price **$59.95**

Radiation Monitor Alarm Kit

This is an inexpensive surface-mount project that is good for beginners to start with. This kit has its own printed circuit board (PCB) which makes mounting the components easy. Plus, it comes in a pocket size shielded aluminum case measuring 3.5" in length and 1" in diameter. The on-off switch is a pushbutton on the bottom.
Subscriber's Price **$32.45**
Non-Subscriber's Price **$33.95**

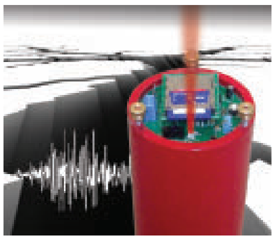
Geiger Counter Kit



As seen in the March 2013 issue.
This kit is a great project for high school and university students. The unit detects and displays levels of radiation, and can detect and display dosage levels as low as one micro-roentgen/hr. The LND712 tube in our kit is capable of measuring alpha, beta, and gamma particles.

Partial kits also available.
Subscriber's Price **$145.95**
Non-Subscriber's Price **$148.95**

Seismograph Kit

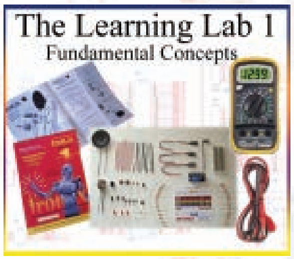


As seen in the May 2012 issue.
Now you can record your own shaking, rattling, and rolling.
The Poor Man's Seismograph is a great project /device to record any movement in an area where you normally shouldn't have any. The kit includes everything needed to build the seismograph. All you need is your PC, SD card, and to download the free software to view the seismic event graph.
Subscriber's Price **$79.95**
Non-Subscriber's Price **$84.95**

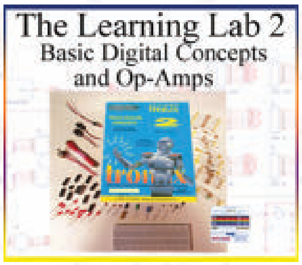
Battery Marvel

As seen in the November 2011 issue.
Battery Marvel helps protect cars, trucks, motorcycles, boats, and any other 12V vehicles from sudden battery failure. This easy-to-build kit features a single LED that glows green, yellow, or red, indicating battery health at a glance. An extra-loud piezo driver alerts you to any problems.
Details, please visit our website.
Subscriber's Price **$18.95**
Non-Subscriber's Price **$19.95**

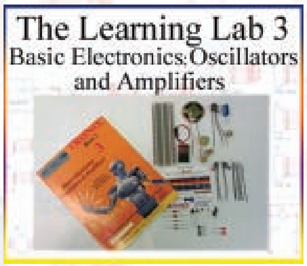
FOR BEGINNER GEEKS!



$59.95



$49.95



$39.95

The labs in this series — from GSS Tech Ed — show simple and interesting experiments and lessons, all done on a solderless circuit board.
As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.
For more info and a promotional video, please visit our webstore.

CLASSIFIEDS

SURPLUS

SURPLUS ELECTRONIC PARTS & ACCESSORIES

Over 20,000 Items in Stock.

Cables, Connectors, Displays, Fans; Hardware, LEDs, Motors, Potentiometers; Relays, Semiconductors, Service Manuals, Speakers; Switches, Test Equipment, Tools, VCR Parts

Surplus Material Components
SMC ELECTRONICS
www.smcelectronics.com

No Minimum Order.
Credit Cards and PAYPAL Accepted.
Flat $4.95 per order USA Shipping.

COMPONENTS

RF PARTS™ CO.
Tubes, Transistors, Power Components
Email: rfp@rfparts.com • Web: www.rfparts.com
800-737-2787 Fax 888-744-1943

MISC FOR SALE

HUGE CLOSE-OUT SALE!
All kinds of electronic products and parts!
Test equipment, schematics, tubes, audio-video equipment ... too numerous to list. Call for availability.
loustvelectronics@verizon.net
215.271.1138 or 215.468.3028
215.432.0333
Call Anytime/24-7

5-Digit LED Display
For the Basic Stamp and other micro-controllers.
Uses the MC14489 chip
$28.99 w/FREE SHIPPING
See our web site for details.
www.boatanchoraudio.com

ROBOTICS

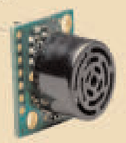
MaxBotix®
High Performance Ultrasonic Rangefinders

Check us out Today!

Save 10% NOW On Web Orders Enter Code: NV31714

HRXL-MaxSonar®-WR™
- High noise tolerance
- IP67 rated
- 1 mm resolution
- Multi-Sensor operation
- Calibrated beam pattern
- Starting at $109.95

XL-MaxSonar®-EZ™
- Great for UAV's and robotics
- Incredible noise immunity
- Small in size
- 1cm resolution
- Automatic calibration
- Starting at $39.95



Phone: 218-454-0766 Email: sales@maxbotix.com
www.maxbotix.com

NEW PRODUCTS

Continued from page 21

- LED stage, studio, and theater lighting.
- Prop and set effects low voltage DC cabling.
- LED decorative lighting.
- Small solar panel systems.
- Portable battery packs.
- Low voltage DC extension power cables and power supply output cable assemblies.
- Equipment interconnect.
- CCTV security camera power extension cables.

Pricing is as follows: 25 feet $18.70; 100 feet $67; 500 feet $300; and 1,000 feet $550.

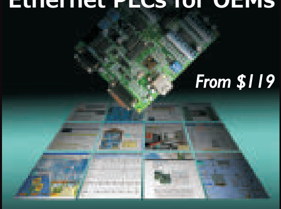
For more information, contact:
J2 LED Lighting
http://j2ledlighting.com

KITS/PLANS

QKITS.COM
We have lots of fun electronic projects for everyone of any skill level!
Stock up now for your winter building!
FREE SHIPPING On all ground orders over $175
1-888-GO-4-KITS

CONTROLLERS

INTEGRATED
Ethernet PLCs for OEMs



NANO-10
- ETHERNET / Modbus TCP/IP
- 8 DI/Os, 2 AI/Os
- RS485
- PWM / PID / Stepper Control
- Ladder + BASIC Programming

tel : 1 877 TRI-PLCS
web : www.triplc.com/nv.htm
TRIANGLE RESEARCH INTERNATIONAL

Nuts & Volts Mobile App!
Check it out now at your favorite App Store!

WANT TO BUY

Excess Solutions
156 S. Milpitas Blvd.
Milpitas, CA 95035
Come in and see our retail store!
We Buy & Sell Excess Electronic Inventory
(408) 262-3900
www.excesssolutions.com

HARDWARE WANTED

DEC EQUIPMENT WANTED!!!
Digital Equipment Corp. and compatibles.
Buy - Sell - Trade
CALL KEYWAYS 937-847-2300
or email buyer@keyways.com

MOBILE SCOOTERS

Sunny Solutions
Sales and Service
Need Help Getting Around?
Elec Mobile Scooters
www.ScootersByTom.com
800 315-0309

LIGHTING

SOLAR

Single and Dual Axis Light Seeking Solar Tracker Controls
www.theanalogguy.com

If you have a new product that you would like us to run in our New Products section, please email a short description (300-500 words) and a photo of your product to:

newproducts@nutsvolts.com

>>> QUESTIONS

Motor Kickback

I am using an Ametek 965922-101 brushed DC motor (rated 38V nominal, 12A peak) in a project that runs it as both a drive and a brake. (Interestingly, the 38V rating is a "bus rating," i.e., with the motor stalled at 38V, it will use 12A and get nothing done.) In drive mode, the controller typically runs it up to 60V; current limited to 10A. Works great.

For the brake, I am using an IRFB260 MOSFET with a 0.6 ohm spring in series to limit the surge current. The PWM rate is 120 Hz. (**NOTE:** When sinking 1/2 HP of energy, the motor generates 30V.) However, each time the MOSFET turns off (at 120 Hz), the motor provides a huge kickback, and the MOSFET's reverse-protection zener diode probably will not survive that for very long.

At first, I paralleled the motor with an Elite 100 µF 400V capacitor [marked PM 105°C, (M 9305)] which calmed down the splash nicely. Then one day after some generous usage, a huge cloud of smoke boiled out of a severely melted capacitor. It was a very abrupt introduction to the (unmarked) ripple current rating on capacitors. At the moment, I'm just using the same 4,700 µF 100V bolt-mounted huge capacitor that filters the drive circuit. (When using the 100 µF capacitor, one of the relay contacts would switch the 4,700 µF capacitor out of the circuit.) However, the 0.64 ohm brake circuit discharges the capacitor so quickly (at 120 Hz) that brake control is either none or a lot.

The challenges are that the kickback is positive, it needs to be kept under 200V, and there is a lot of energy behind the kick from this motor. I ended up with 100 µF in the first place because a large (non-electrolytic) 10 µF capacitor did very little to the splash, which almost immediately would go beyond the 250V rating of the oscilloscope. In my parts bin are some large diodes such as DTV32, STPR1020CT (2), STPR2045CT (2), TYN058, and others, if that helps. I've read a little about snubber circuits, but would like some technical advice.

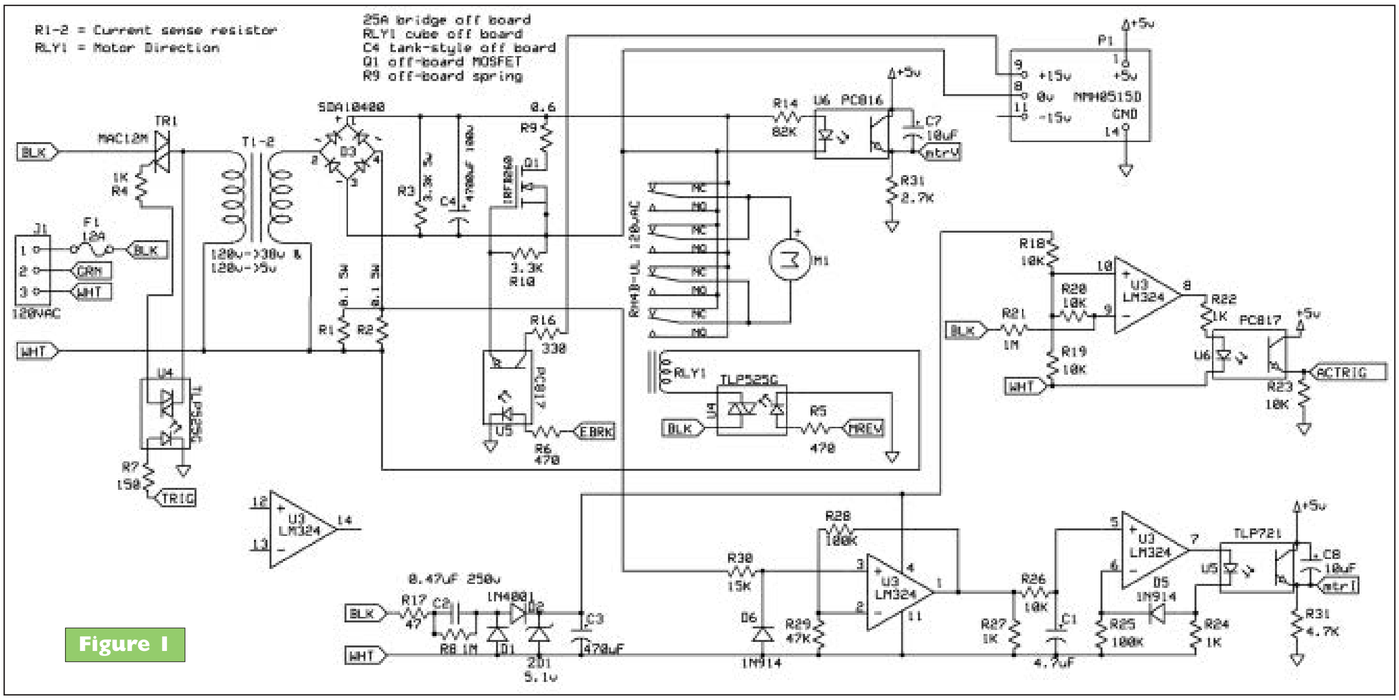


Figure 1

Send all questions and answers by email to **forum@nutsvolts.com** or via the online form at **www.nutsvolts.com/tech-forum**

All questions *AND* answers are submitted by *Nuts & Volts* readers and are intended to promote the exchange of ideas and provide assistance for solving technical problems. All submissions are subject to editing and will be published on a space available basis if deemed suitable by the publisher. Answers are submitted by readers and **NO GUARANTEES WHATSOEVER** are made by the publisher. The implementation of any answer printed in this column may require varying degrees of technical experience and should only be attempted by qualified individuals.

Always use common sense and good judgment!

Send all questions and answers by email to **forum@nutsvolts.com**
or via the online form at **www.nutsvolts.com/tech-forum**

On the schematic in **Figure 1**, the +5V and ground points belong to an isolated 5V 1A cell phone charger. Not shown is the PIC16F747 and associated circuitry that handles the controlling.

#3141 **Tsidqah**
via email

Capacitor Forming

I pulled some excellent quality electrolytic capacitors from the power supply of an amplifier. The capacitors are rated at 40 VDC but were used in a 10V circuit. I wanted to use the capacitors in a 24 VDC circuit, but I was told that the capacitors "formed" at 10V and wouldn't work at 24 VDC, regardless of the original rating. Is this true?

#3142 **Jacob McClure**
Tallahassee, FL

Sweep Generator Range

I have a Heathkit sweep generator 1274 which I bought with the hope I could quickly determine the frequency response of amplifiers I play with from time to time. However, there is no way to determine the range of the pattern on my scope — at least none that I have figured out. I'm not very advanced in such matters. Is there any easy way to add blips to the pattern say at 12 kHz and or 15 kHz?

#3143 **Alan May**
via email

>>> ANSWERS

[#12134 - December 2013]
Fan Noise

My latest creation, a 50W tube amp, produces great, audiophile-quality sound. Problem is, I have two fans cooling the amp, and the noise is distracting. Is there an alternative to forced-air cooling of audio tubes? I've looked at the water cooled systems used for computers, but I don't know if the tubes are too hot for a DIY water jacket.

Back in the day when there were ONLY tube amps, none of them used fans. Hot air rises, and tubes make the air hot enough that convection cooling is all that is needed as long as you properly ventilate the box containing the amplifier. This means long slits or a large amount of nice sized holes in the bottom and the top of the box to let the air flow.

Phil Karras
via email

[#1145 - January 2014]
Schematics Needed

I found an old transistor radio in my parent's attic. It's one of the first transistor radios made by RCA. Is there a good source for schematics on old radios and other electronics? I've tried the usual Google searches, but turned up nothing.

#1 Rider schematics were published up to 1954 — too early to have included your radio which was probably made in the early 1960s. Howard W. Sams photofact service folders started in 1946 and may still be in production. I have an index which I believe I downloaded from **www.servicesoftware.com** in 2010. It lists 16,283 RCA schematics of radios, TVs, record players, and the like. So, if a schematic of your radio exists, it should be there. All you need is the model number to find it. I don't know the cost to download a schematic but it is not free. You can also check **www.samswebsite.com**.

Russell Kincaid
Milford, NH

#2 Whether it's an old radio, TV, test instrument, audio gear, anything "antique" — including older transistor radios — go to **www.antiqueradios.com** and check out their forum, especially under the section "transistor radios." This is the best source for help with antique electronics anywhere on the Internet.

Dean Huster
Harviell, MO

#3 Have you tried SAMS Photofact at **www.theschematicman.com**? They probably had one for that radio; it's in he correct time frame. You'll need the make (RCA) and model number; perhaps even a version number on the circuit board. I don't know about radios, but it did help when I was working on TVs.

Phil Karras
via email

[#1146 - January 2014]
Telescope Position Control

I'm looking for a motor drive for my 6" refractor telescope. A stepper motor should give me position control, but big steppers are expensive. DC motors are cheaper, but require a complex gear box and a sensor to determine position. Is there an affordable option that provides me with the positioning benefits of a stepper? My telescope and camera back weigh about 6 lbs total.

#1 Telescope position control is not for the faint at heart. First of all, the controller must be able to move the telescope in Azimuth (in a circle around the scope) and elevation (vertical angle from the horizon to the point above the scope — zenith), and convert these angles to the celestial equivalents of latitude and longitude called declination and right ascension, respectively.

You must start with the telescope aligned with the Earth's axis of rotation which can be accomplished by sighting Polaris (the North Star). Then, the telescope position controller must be able to track the object you are interested in by countering the Earth's rotational motion.

Before tackling building a positioning controller (not an easy feat involving electronics, mechanical systems, and programming), look at

Send all questions and answers by email to **forum@nutsvolts.com** or via the online form at **www.nutsvolts.com/tech-forum**

Celestron (**www.celestron.com**), Meade (**www.meade.com**), or Gemini (**www.bisque.com/help/theskyv6/telescope/Gemini_by_Losmandy_Instruments.htm**). These companies make telescope positioners for a variety of telescope sizes and types.

Tim Brown PhD EE, PE
via email

#2 Why would you use a big stepper to drive a telescope? Surely you are not considering a direct connection to the telescope's polar axis? That would produce a very jerky motion instead of the smooth motion needed for observation or photography. Therefore, you still need gears and you really need a high ratio worm gear or a series of compound gears to get a high reduction ratio. A relatively small stepper or DC motor will then work just fine.

Amateur telescope makers have used a wide variety of drive mechanisms. I have seen worm drives made by wrapping a threaded rod around a wood disk to form the "teeth." A screw with the same thread pitch can be used to drive it. If the scope is well balanced, little torque is needed to drive it. On the other hand, second hand worm wheels can often be found in places like eBay and other resellers.

Since a telescope moves at such a slow rate, some are driven by a cord or wire or strip of metal that is wrapped around a wheel instead of a gear. The cord/wire/strip is pulled by a nut that runs on a threaded rod that is turned by the motor. Since you can track from sunset to sunrise with only a half turn of the polar axis, you don't even need the full circle on that wheel. A fast mode or a nut that can be released would allow faster resetting for another observation.

Paul Alciatore
Beaumont, TX

[#2141 - February 2014]
Traffic Detector

I live in the country on a side road that tee's into a busy arterial. The arterial curves sharply to the left, offering only about 100 feet of visibility. Most of my trips involve turning left, thus crossing one lane of oncoming traffic. A car coming around the curve at 60 MPH gives me a little over one second reaction time when it first becomes visible.

At night, I can see oncoming headlights reflecting off a guardrail which gives me plenty of warning. During the day, I roll down my window and listen. However, this is not the best method with my aging hearing. I'm looking for a clever electronic solution to detect approaching cars and provide an earlier warning.

There is a pole on the other side of the road about 75 feet away that could be used to mount a device to bounce a signal off of.

#1 How about mounting a box with a Doppler radar module (eBay search "Doppler radar module") mounted on the pole across the road in a plastic box with an LED to indicate when the Doppler detects a car? The modules cost under $10 from the far east. Biggest problem I can see is providing power. If mains power is not available, you could use a solar panel and rechargeable battery. You need to check local laws regarding possible licensing of the Doppler transmitter and get permission of the pole's owner.

Robert Atkinson
Cambridge, UK

#2 Robert beat me to the punch regarding the Doppler. My thought was to install both a Doppler sensor (like a Solfan unit) and a small Part 15 FM transmitter on the pole. The FM transmitter can be directly modulated by the sensors detector, and may be far more sensitive than the sensor's relay output. You can also hear the Doppler shift and get an idea of direction and speed of traffic in the sensor's view. Pick a clear spot on your car radio and preset it to hear your detector. You will need a bit of power to run all this — especially the sensor if it has a gunn diode — and you will need to calculate the current required to run the circuits and charge a battery. If the Doppler sensor uses an inefficient regulator, consider a switching regulator.

Joe Leikhim
via TF Comment

#3 You might want to look at using a microwave Doppler sensor which you can find at **DX.com** searching for: Jtron HB100 10.525 GHz Microwave Doppler Wireless Radar/Detector Probe Sensor. It is less than $10.

A pair could be used to detect vehicles approaching from either direction. The sensor could have a solar panel charging a battery, a low power receiver which you would trigger with a transmitter when you approach the intersection, which then would turn on the two radar transmitters; one first, then the other. If traffic is sensed, the unit could flash a bright red LED, otherwise flash a green LED and time out after a minute or so.

There is no need to be transmitting radar signals all the time; this wastes battery power. Powering a receiver which then turns on the Doppler sensors offers much less constant power draw, thereby making the power demands less challenging. For the RF link, one could hack a receiver/transmitter that is intended as a mailbox monitor or driveway monitor. Each has a receiver circuit that has modest power draw and are available for a reasonable cost.

Arlen Raasch
Fredericksburg, VA

#4 Contact your local road department. Around here in SE KY, they put up fairly large convex mirrors for just the problem you asked about.

B Jim Russell N4ICU
KY

#5 I see a couple of problems with your idea of a Traffic Detector. First of all, the "pole on the other side of the road" belongs to a utility company and they are usually pretty adamant about "devices" being installed on their poles (for years, my father tried to wrap a pole with sheets of 16 gauge aluminum to keep squirrels out of a pecan tree and the power company had him remove the sheets pronto). If the company allows you to put up a target to bounce a signal from, there is still the problem of aligning the target such that any size vehicle would be picked up EVERY time (or your safety would be compromised by a missed vehicle). Using sonar to bounce off a target most likely would not work, and using radar would get into issues with the FCC, plus the need for Doppler signal processing to ensure you detect only moving vehicles and not plants or stationary devices (a.k.a., very complicated digital signal processing and VERY costly).

Since you are using your hearing now to detect approaching vehicles, why not use a Super Ear listening device ($29.50 at **www.amazon.com/SuperEar-Personal-Amplifier-Listening-Compliance/dp/B000X2H8G4/ref=pd_sim_hi_1/182-4508871-5045937**) or the Bionic Ear and Booster set for even better results ($155.99 at **www.amazon.com/Bionic-Ear-And-Booster-Set/dp/B0012N6GZ2**).

Building either of these devices would require lots of electronic/construction skills (not for the novice since the Super Ear uses a highly directional microphone and the Bionic Ear uses a parabolic reflector); adjustment/alignment of the device to ensure accurate identification of approaching vehicles would be critical; and the cost of construction more than likely would equal or exceed the cost of a proven manufactured device. I am an old school electronics person and love building, but when safety is a concern it is best to use a proven device rather than a home brew.

Tim Brown PhD EE, PE
via email

■ LOOK FOR ■ SEARCH FOR ■ FIND

Find your favorite advertisers here!

ADvertiser INDEX

IT'S SPRINGTIME AT RAMSEY!

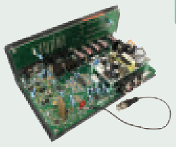
Super-Pro FM Stereo Radio Station

- ✔ PLL synthesized for drift-free operation
- ✔ Built-in mixer - 2 line inputs and one microphone input, line level monitor output!
- ✔ Frequency range 88.0 to 108.0, 100 kHz steps
- ✔ Precision active low-pass "brick wall" audio filter!
- ✔ Dual LED bar graph audio level meters!
- ✔ Automatic adjustable microphone ducking!
- ✔ Easy to build through-hole design!

This professional synthesized transmitter is adjustable directly from the front panel with a large LED digital readout of the operating frequency. Just enter the setup mode and set your frequency. Once selected and locked you are assured of a rock stable carrier with zero drift. The power output is continuously adjustable throughout the power range of the model selected. In addition, a new layer of anti-static protection for the final RF amplifier stage and audio inputs has been added to protect you from sudden static and power surges.

Audio quality is equally impressive. A precision active low-pass brick wall audio filter and peak level limiters give your signal maximum "punch" while preventing overmodulation. Two sets of rear panel stereo line level inputs are provided with front panel level control for both. Standard unbalanced "RCA" line inputs are used to make it simple to connect to the audio output of your computer, MP3 player, DVD player, cassette deck or any other consumer audio source. Get even more creative and use our K8094 below for digital storage and playback of short announcements and ID's! In addition to the line level inputs, there is a separate front panel microphone input. All three inputs have independent level controls eliminating the need for a separate audio mixer! Just pot-up the source control when ready, and cross fade to the 2nd line input or mic! It's that simple! In addition to the dual stereo line inputs, a stereo monitor output is provided. This is perfect to drive studio monitors or local in-house PA systems.

The FM100B series includes an attractive metal case, whip antenna and built in 110/220VAC power supply. A BNC connector is also provided for an external antenna. Check out our Tru-Match FM antenna kit, for the perfect mate to the FM100B transmitter.



We also offer a high power kit as well as an export-only assembled version that provides a variable RF output power up to 1 watt. The 1 watt unit must utilize an external antenna properly matched to the operating frequency to maintain a proper VSWR to protect the transmitter.

(Note: The FM100B and FM100BEX are do-it-yourself learning kits that you assemble. The end user is responsible for complying with all FCC rules & regulations within the US or any regulations of their respective governing body. The FM100BWT is for export use and can only be shipped to locations outside the continental US, valid APO/FPO addresses or valid customs brokers for documented end delivery outside the continental US).

FM100B	**Super-Pro FM Stereo Radio Station Kit, 5uW to 25mW Output**	SPRING SALE! ***$239.95***
FM100BEX	**Super-Pro FM Stereo Radio Station Kit, 5uW to 1W Output**	***$299.95***

Four-Mode Keyless Entry Test Set

Just like the days of "plugs, points, and condenser" are over, so are the days of having the hardware store grind out a spare key for your car!

Now when your keyless access system doesn't work, you need to accurately detect what part of the system is malfunctioning. This could be anything from a dead battery in the key fob, a "brain-dead" key fob, to malfunctioning sensors, antennas, or other system components in the vehicle. Until now there was no way to determine where the system was failing.

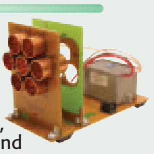
Testing your system is easy. To test the complete 125 kHz/315 MHz communications path just stand close to the vehicle with the WCT3 and your key fob in hand. Press the test button and the WCT3 will detect and display the presence of the vehicle's 125kHz/20KHz signal and, if they "handshake", will also detect and display the presence of your key fob's 315MHz return signal. You can independently test key fob only signals (panic, lock, trunk, etc.) by holding the key fob near the WCT3, pressing the test button, and pushing the function button on the key fob.

The same functionality testing can be done with IR key fobs. The modulated IR signal is detected and will illuminate the IR test LED on the test set.

If you know a few "secrets" you can also see if the tire pressure sensors/transmitters are generating signals or the built-in garage door opener in your rear view mirror is transmitting a signal! Runs on a standard 9V battery. Also available factory assembled & tested.

WCT3 Keyless Entry Test Set Kit $59.95

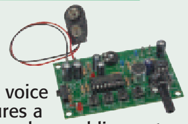
Air Blasting Ion Generator



Generates negative ions along with a hefty blast of fresh air, all without any noise! The steady state DC voltage generates 7.5kV DC negative at 400uA, and that's LOTS of ions! Includes 7 wind tubes for max air! Runs on 12-15VDC.

IG7 Ion Generator Kit $64.95

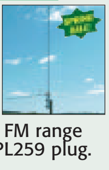
Digital Voice Changer



This voice changer kit is a riot! Just like the expensive units you hear the DJ's use, it changes your voice with a multitude of effects! Features a built-in microphone and both a speaker and line output. Runs on a standard 9V battery, not included.

MK171 Digital Voice Changer Kit $14.95

Collinear Vertical FM Antenna



Our 5/8 wave omni antenna has been the standard for LPFM insallations worldwide. Provides 3.4dB gain while keeping the signal radiation low to the horizon for maximum range. Field tuneable over the entire FM range for a perfect match. SO239 connector for PL259 plug.

FMA200E Omnidirectional FM Antenna *$119.95*

HV Plasma Generator

Generate 2" sparks to a handheld screwdriver! Light fluorescent tubes without wires! This plasma generator creates up to 25kV at 20kHz from a solid state circuit! Build plasma bulbs from regular bulbs and more! Runs on 16VAC or 5-24VDC.

PG13 HV Plasma Generator Kit $64.95

Audio Recorder & Player

Record and playback up to 8 minutes of messages from this little board! Built-in condenser mic plus line input, line & speaker outputs. Adjustable sample rate for recording quality. 4-switch operation that can be remote controlled! Runs on 9-12VDC at 500mA.

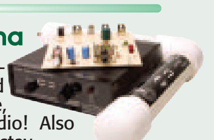
K8094 Audio Recorder/Player Kit $32.95

Logic Interface Module

Interface your digital output to the real world with an on-board SPDT relay rated at 240V at 10A! It takes a digital low (.5VDC or less) or a high (+1 to +12VDC) and provides your choice of an active low or high closure! It's that simple! Runs on 12VDC at 60mA.

RI1 Logic Interface Kit $17.95

Signal Magnet Antenna



The impossible AM radio antenna that pulls in the stations and removes the noise, interference, and static crashes from your radio! Also helps that pesky HD AM Radio stay locked! Also available factory assembled.

SM100 Signal Magnet Antenna Kit $89.95

Passive Aircraft Monitor

PATENTED!

The hit of the decade! Our patented receiver hears the entire aircraft band without any tuning! Passive design has no LO, therefore can be used on board aircraft! Perfect for airshows, hears the active traffic as it happens! Available kit or factory assembled.

ABM1 Passive Aircraft Receiver Kit $89.95

4-Channel USB Relay Control

This professional quality USB relay controller allows computer controlled switching of external devices, plus full bi-directional communication with the external world using the USB port of your computer!

The controller features four onboard relay outputs with a current rating of 10A each. Also onboard is a 6-channel Input/Output interface, with each channel individually configurable as Digital Input, Digital Output, Analog Input (10-bit Resolution). In Digital Input/Output modes, each channel can support a TTL compatible or ST input or a 5V output signal. In Analog Input mode, each channel can convert a voltage of between 0 to 5V into a 10-bit digital representation.

UK1104 4-Ch USB Relay Interface Kit $59.95

Broadband RF Preamp

Need to "perk-up" your counter or other equipment to read weak signals? This preamp has low noise and yet provides 25dB gain from 1MHz to well over 1GHz. Output can reach 100mW! Runs on 12 volts AC or DC or the included 110VAC PS. Assmb.

PR2 Broadband RF Preamp $69.95

Laser Trip Sensor Alarm

True laser protects over 500 yards! At last within the reach of the hobbyist, this neat kit uses a standard laser pointer (included) to provide both audible and visual alert of a broken path. 5A relay makes it simple to interface! Breakaway board to separate sections.

LTS1 Laser Trip Sensor Alarm Kit $29.95

Active Receive Antenna

The popular antenna for the serious DX'ers works on all bands - shortwave, HF, VHF, and UHF yet performs like a 60' long wire antenna! Provides over 15dB of gain, and includes auto-off RF bypass and front panel gain control.

AA7C Active Antenna Kit $59.95

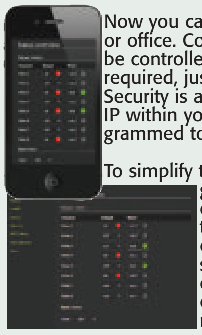
ramsey
www.ramseykits.com

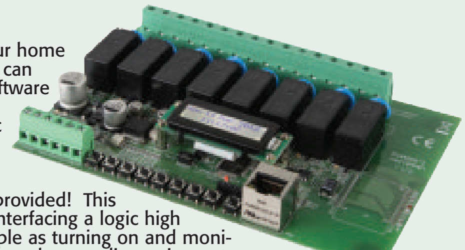
There's only so much room on these two pages, so check it all out in our new virtual electronic catalog! Flip through the pages and search with ease! Visit ***www.ramseycatalog.com***

Follow Us and SAVE $$

Follow us on your favorite network site and look for a lot of super deals posted frequently... exclusively for our followers!

8-Channel Remote Ethernet Controller





Now you can easily control and monitor up to 8 separate circuits via the standard Ethernet network in your home or office. Connection wise it couldn't be simpler. The controller functions as an IP based web server, so it can be controlled by any internet browser that can reach your network! There are no drivers or proprietary software required, just access the controller like any web page from your PC, laptop, or even your smartphone! Security is assured allowing up to 4 separate user credentials. The controller can be set to a specific static IP within your network subnet or can be set to DHCP (auto negotiate). The controller can even be programmed to send you an email to notify and confirm power up and status changes!

To simplify the connection of your equipment to the controller, 8 separate and isolated relay outputs are provided! This gives you internet or network control of up to 8 separate functions. No need to worry about interfacing a logic high or logic low, or burning up the interface! The applications are endless! From something as simple as turning on and monitoring lights at your house with a normal latched closure to advanced control of your electronic gadgets, radio equipment, or even your garage door! Each relay contact is rated at 12A at 30VDC or 16A at 230VAC. Each of the 8 channels has built-in timer and scheduler programs for day, weekend, working days, every day, and every day except Sunday. Relay control functions are programmable for on, off, or pulse (1-99 seconds, 1-99 minutes, or 1-99 hours). In addition to control functions, the web interface also displays and confirms the status of each channel. Each channel can be custom labeled to your specific function name. The controller operates on 12VDC or 12VAC at 500mA or our new AC121 global 12VDC switching power supply below. Factory assembled, tested, and ready to go! Even includes a Cat-5 cable!

VM201 8-Channel Remote Ethernet Controller, Factory Assembled & Tested $169.95

Laser Beam Audio Communicator

Now you can talk to your friends over one of the most secure long-distance transmission types available, a laser beam! The transmitter uses a microphone or external audio to modulate a laser beam on and off at a rate of more than 16kHz so the audio fidelity is much better than that of a standard 3kHz telephone line!

The receiver includes filtering to remove the 16kHz carrier and leave behind the high quality audio, and then boost its level for use with earphones. Transmitter audio AGC keeps your level perfect! Includes transmitter, receiver and laser pointer. Each runs on a 9V battery (not included).

LBC6K Laser Beam Audio Communicator Kit $59.95

Precision PC Plane Antennas

Our LPY series PC antennas continue to be the favorite for virtually all RF and wireless applications. From microwave links, wireless mics, to RFID, we've got you covered. Check our site for details!

LPYSeries Precision PC Plane Antennas from $29.95

Stereo Ear Super Audio Amp

This "Stereo Ear" amp is one of the neatest and handiest high gain amps you will find! Dual high sensitivity electret mics are amplified 50x to provide the ultimate stereo source! Output is 3.5mm jack, runs on 3ea AAA's.

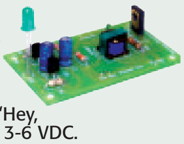
MK136 Stereo Ear Audio Amplifier Kit $9.95

Tone Encoder/Decoder

Encode and decode with the same kit! This little mini-kit will simultaneously encode and/or decode any audio frequency between 40Hz and 5,000 Hz! Precision 20-turn trim pot adjustment! 5-12VDC.

TD1 Tone Encoder/Decoder Kit $9.95

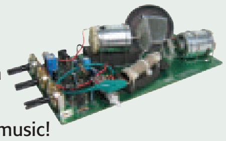
Tickle-Stick Shocker



The kit has a pulsing 80 volt tickle output and a mischievous blinking LED. And who can resist a blinking light and an unlabeled switch! Great fun for your desk, "Hey, I told you not to touch!" Runs on 3-6 VDC.

TS4 Tickle Stick Kit $9.95

Laser Light Show



Just like the big concerts, you can impress your friends with your own laser light show! Audio input modulates the laser display to your favorite music! Adjustable pattern & speed. Runs on 6-12VDC.

LLS1 Laser Light Show Kit $49.95

Optically Isolated Module

The hobbyist's headache solver! Converts any AC or DC signal to logic level. The beauty is that the input and output are totally isolated from each other! Output can drive up to 150mA at 40VDC.

OM2 Optically Isolated Module Kit $16.95

High Intensity SMT Blinky

Want to learn working with and soldering SMT components? No better way then our SMT Blinky kit! Learn the fun way and end up with a subminiature high intensity LED alternating display! Includes 2ea LR55 button cells.

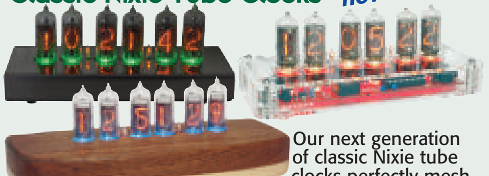
BL2 High Intensity SMT Blinky Kit $17.95

12VDC Regulated

Go green with our new 12VDC 1A regulated supply. Worldwide input 100-240VAC with a Level-V efficiency! It gets even better, includes DUAL ferrite cores for RF and EMI sup-

AC121 12VDC 1A

Classic Nixie Tube Clocks *HOT SELLER!*



Our next generation of classic Nixie tube clocks perfectly mesh today's technology with the Nixie era technology of the 60's. Of course, features you'd expect with a typical clock are all supported with the Nixie clock... and a whole lot more!

The clocks are programmable for 12 or 24 hour mode, various AM/PM indications, programmable leading zero blanking, and include a programmable alarm with snooze as well as date display, 4 or 6 tube, kit or assembled!

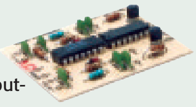
We then jumped the technological time line of the 60's Nixie displays by adding the latest multi-colored LEDs to the base of the Nixie tubes to provide hundreds of illumination colors to highlight the glass tubes! The LED lighting can be programmed to any color and brightness combination of the colors red, green, or blue to suit your mood or environment.

Then we leaped over the technological time line by integrating an optional GPS time base reference for the ultimate in clock accuracy! The small optional GPS receiver module is factory assembled and tested, and plugs directly into the back of the clock to give your Nixie clock accuracy you could only dream of!

The clocks are available in our signature hand rubbed Teak & Maple, polished stainless, or clear acrylic bases. You also have your choice of IN-14 or highly sought after IN-8-2 nixie tubes (for the 6-tube clock).

NIXIE Classic Nixie Tube Clock Kits From $229.95

Touch Switch



The ultimate touch switch! Touch once - it's on, touch again - it's off, or use the momentary outputs that stay on only as long as touched. Two switch circuits on each board. Drives loads up to 100mA. Runs on 6-12VDC.

TS1 Touch Switch Kit $9.95

12VDC Worldwide

It gets even better than our AC121 above! Now, take the regulated Level-V green supply, bump the current up to 1.25A, and include multiple blades for global country compatibility!

PS29 12VDC 1.25A Global Power Supply $19.95

Tri-Field Meter Kit

"See" electrical, magnetic, and RF fields as a graphical LED display on the front panel! Use it to detect these fields in your house, find RF sources, you name it. Featured on CBS's Ghost Whisperer to detect the presence of

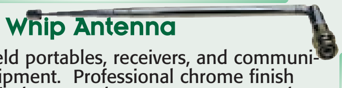
TFM3C Tri-Field Meter Kit $74.95

Electronic Watch Dog

A barking dog on a PC board! And you don't have to feed it! Generates 2 different selectable barking dog sounds. Plus a built-in mic senses noise and can be set to bark when it hears it! Adjustable sensitivity! Unlike my Greyhound, eats 2-8VAC or 9-12VDC, not fussy!

K2655 Electronic Watch Dog Kit $39.95

Telescopic Whip Antenna



Ideal for handheld portables, receivers, and communications test equipment. Professional chrome finish with integral swivel mounted BNC connector matche virtually any application whether top mounted or 90 degree rear panel mounted. Whip extends 5.25" to 27.75" to cover a wide variety of frequencies.

WA10 Telescopic Whip Antenna, BNC $14.95

Sniff-It RF Detector Probe

Measure RF with your standard DMM or VOM! This extremely sensitive RF detector probe connects to any voltmeter and allows you to measure RF from 100kHz to over 1GHz! So sensitive it can be used as a RF field strength meter!

RF1 Sniff-It RF Detector Probe Kit $27.95

GET THE NUTS AND VOLTS DISCOUNT!

*Mention or enter the coupon code **NVRMZ142** and receive 10% off your order!*

800-446-2295

www.ramseykits.com

Prices, availability, and specifications are subject to change. Not responsible for typos, mistakes, stupids, or confusion that Spring is even approaching! Robin gave me an extra week to dig out of the snow and prepare this Spring ad! How ironic is that! Visit www.ramseykits.com for the latest pricing, specials, terms and conditions. Copyright 2014 Ramsey Electronics®.. so there!

RAMSEY ELECTRONICS®
590 Fishers Station Drive
Victor, NY 14564
(800) 446-2295
(585) 924-4560

SDS7102 100MHz 1GSa/s 10Mpt 2-Channel Oscilloscope

$429

Deep Memory - Battery Option!

The Owon SDS7102 is a 100MHz bandwidth scope with huge 10Mpt memory for fast signal investigations. Battery option for use away from 110V - great for field work!

"I never expected a high end scope for around $400 but Owon delivers more features than I anticipated." -N.

"For the price this scope cannot be beat!" -D.

Your choice!

2-Channel Digital Storage Oscilloscopes with High-End Features:

High-performance, economic digital oscilloscopes for education, training, production line, and R&D. 8-inch color 800x600 TFT_LCD screen, AutoScale, function to simplify use. DB-15 VGA port shows screen display on external monitors or video projectors - ideal for teaching. USB flash disk storage, Pass/-Fail, LAN for remote measurements. Sophisticated triggering includes: Edge, Pulse, Video, Slope. Waveform recording and replay. Comes with probes, USB cable, probe adjust pen, software CD, power adapter, and user manual.

5% Educational Discount Call for details!

FREE Ground Shipping

Free Case

Saelig.com unique electronics

www.saelig.com 888-7SAELIG

SDS5032E 30MHz 250MSa/s 2-Channel Oscilloscope

$289

Lowest Cost Scope!

The Owon SDS5032E is an economical 2-channel 30MHz 250MSa/s digital storage oscilloscope with 10Kpt memory. Excellent starter scope!

"Great scope for the price. Love this scope, especially the clear full color 800x600 screen." -A.

"...nice variety of features and unmatched for price. " -Z.